DTIC FILE COPY

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

AD-A217 377

1b. RESTRICTIVE MARKINGS: NONE

3. DISTRIBUTION/AVAILABILITY OF REPORT: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S): AFIT/CI/CIA-89-030

6a. NAME OF PERFORMING ORGANIZATION: AFIT STUDENT AT Stanford UNIV

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION: AFIT/CIA

6c. ADDRESS (City, State, and ZIP Code)

7b. ADDRESS (City, State, and ZIP Code): Wright-Patterson AFB OH 45433-6583

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification) (UNCLASSIFIED)
MADALINE RULE II: A NEW METHOD FOR TRAINING NETWORKS OF ADALINES

12. PERSONAL AUTHOR(S): Rodney G. Winter

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
THESIS/DISSERTATION	FROM _____ TO _____	1989	102

16. SUPPLEMENTARY NOTATION: APPROVED FOR PUBLIC RELEASE IAW AFR 190-1
ERNEST A. HAYGOOD, 1st Lt, USAF
Executive Officer, Civilian Institution Programs

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

DTIC ELECTE FEB 01 1990 S D

90 02 01 012

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
ERNEST A. HAYGOOD, 1st Lt, USAF	(513) 255-2259	AFIT/CI

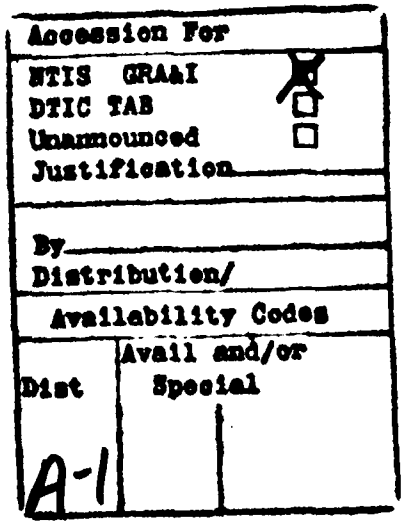
DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

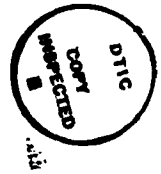
AFIT/CI "OVERPRINT"

MADALINE RULE II: A NEW METHOD FOR TRAINING NETWORKS OF ADALINES

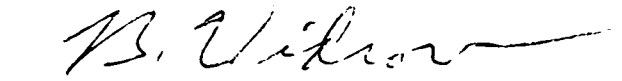
A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Rodney G. Winter
January 1989



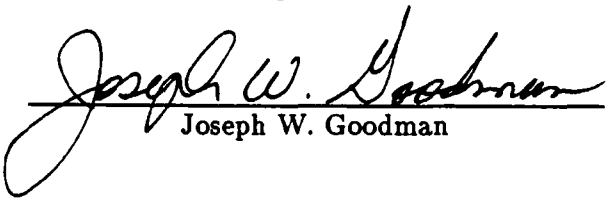


I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



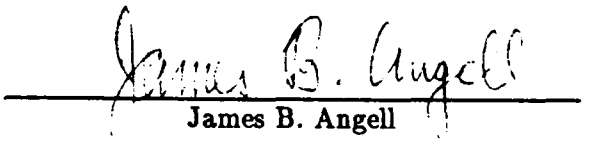
Bernard Widrow
(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



Joseph W. Goodman

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



James B. Angell

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies

Abstract

Madaline Rule II, MRII, is a supervised learning algorithm for training layered feed-forward networks of Adalines. An Adaline is a basic neuron-like processing element which forms a binary output determined by a weighted sum of its inputs. The algorithm is based on a "minimal disturbance" principle. This principle states that changes made to the network's weights in order to correct erroneous responses for a particular input pattern should disturb the responses to other input patterns as little as possible. MRII uses a sequence of trial adaptations to correct output errors during training. The trials that require the smallest weight changes are performed first. A method to insure that all neural elements share responsibility for forming the global solution is introduced and called "usage."

The algorithm exhibits interesting generalization properties. Generalization is the network's ability to make correct responses to inputs not included in the training set. Networks that contain more Adalines than necessary to solve a given training problem exhibit good generalization when trained by MRII on a sufficiently large training set.

The algorithm does not always converge to known solutions. A training failure sometimes occurs when a fraction of the output units achieves an early solution. Convergence is blocked because the changes of the hidden pattern set that are needed for a global solution, are incompatible with the partial solution already formed.

The sensitivity of an Adaline to two types of disturbances, random changes in weights and random input errors, is investigated. The percentage change of the Adaline's input/output mapping due to weight changes equals the percentage change in the weights divided by π. The corresponding result holds for input errors. For Madaline networks where the ideal solution corresponds to individual weights selected independently from a random distribution, the Adaline results are extended to predict the sensitivity of the entire network.

Preface

The guidance and direction of Dr. Bernard Widrow during the conduct of this research is gratefully acknowledged. His insistence for simplicity and clarity forced me to arrive at simple results. He is largely responsible for the utility of the results.

The support of Ben Passarelli of Alliant Computer Systems Corporation is also greatly appreciated. Ben made available his company's equipment for performing many of the simulations presented in this report. More importantly, he gave selflessly of his time to teach me how to best use the equipment.

My research group enjoys the support and encouragement of Thomson CSF. Thomson's dedication to the free exchange of information and ideas between academia and industry is exemplary. I acknowledge the efforts of Pierre Martineau and Fred Fisher in setting up a mutually beneficial relationship between our groups.

My colleagues in the "zoo" have been the source of much inspiration. Special thanks goes to Maryhelen Stevenson who proofread much of this report before the reading committee. Her competence saved me some embarrassment.

During the period of this research, the author was an active duty officer in the United States Air Force. He was assigned to Stanford University under the Civilian Institutes Program of the Air Force Institute of Technology. No information or statement contained in this report should be construed to be supported by or be representative of the policies or views of the United States Air Force.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Research in the field of neural networks has seen a resurgence of interest in the past several years. People hope to be able to teach a machine how to perform a given task by merely showing it examples of desired behavior. Since the machine learns by experience, the task at hand does not need to be decomposed into an algorithm that can be programmed into the machine. In this dissertation, a method by which a particular type of neural network can be trained is investigated.

The neural network to be trained is a layered feed-forward Adaline network. The term Adaline, which stands for "adaptive linear neuron," was coined by Widrow in 1959 [1]. The original work presented in this report builds on the work done by Widrow and many of his graduate students in the early 1960s. This introduction reviews some of the past work on Adalines and simple networks of Adalines, called Madalines for "many Adalines." The concepts behind Madaline Rule II, which is a method of training more complicated networks of Adalines, are also presented.

1.1 Contributions

This dissertation presents two major contributions. The first major contribution is the Madaline Rule II algorithm, MRII. The second major contribution is the sensitivity results presented in Chapter 5. There are some minor contributions within the scope of these two major contributions.

The Madaline Rule II, as the "II" suggests, is the result of continuing work started by others. This introductory chapter outlines the previous work. First, the Adaline is

presented. Then, a simple network of Adalines, called a Madaline, is presented. This early Madaline has a much simpler structure than that addressed by Madaline Rule II. The procedures for training these two simple structures are grounded in a concept called "minimal disturbance." Minimal disturbance is one of the basic concepts behind MRII and is the author's inheritance from the early researchers.

The last section of this chapter explains a concept for training a complex Madaline using a sequence of trial adaptations. MRII was developed jointly by the author and Widrow. The details presented in the last section of this chapter are attributable to Widrow. Those details are an expansion of concepts he presented in 1962 [2]. The author's contributions begin in the following chapter.

The author has taken certain ideas of the early researchers and made them work. In formulating MRII, some changes to the minimal disturbance principle are required. These changes are detailed in Chapter 2 and are the author's contributions. Real neural networks need to be built in circuitry. This requirement impacts the design details of the MRII algorithm. The minimal disturbance idea also needs modification to insure all units share responsibility for arriving at a network solution. The modification is implemented by the author as "usage" in the MRII algorithm. After discovering the need for, and implementing usage, the author discovered a similar idea had been proposed by another researcher [3]. The author's experimental techniques for testing the algorithm and the results obtained are contributions. In addition, several heuristics are presented for using the algorithm which are novel and significant.

In working with MRII, the author discovered a failure mode of the algorithm. This mode prevents the algorithm from coming to a solution even when a known solution exists. This mode takes the form of a limit cycle and is detailed in Chapter 4. This discovery is important since an understanding of the phenomenon is essential to improving the algorithm. An attempt to improve MRII by finding an escape from the failure mode led the author to his second major contribution.

The sensitivity of the input/output response of Madalines to errors or drift in the weights of the system as well as to errors in the binary inputs to the system is analyzed. Approximations are made to reduce the analytic results to useful form. These approximations are reasonable, as verified by experiment. The result is a useful and relatively easy to use set of equations that accurately predicts the worst-case sensitivity of Madalines. The result should also be a useful tool for further research into improving the MRII algorithm.

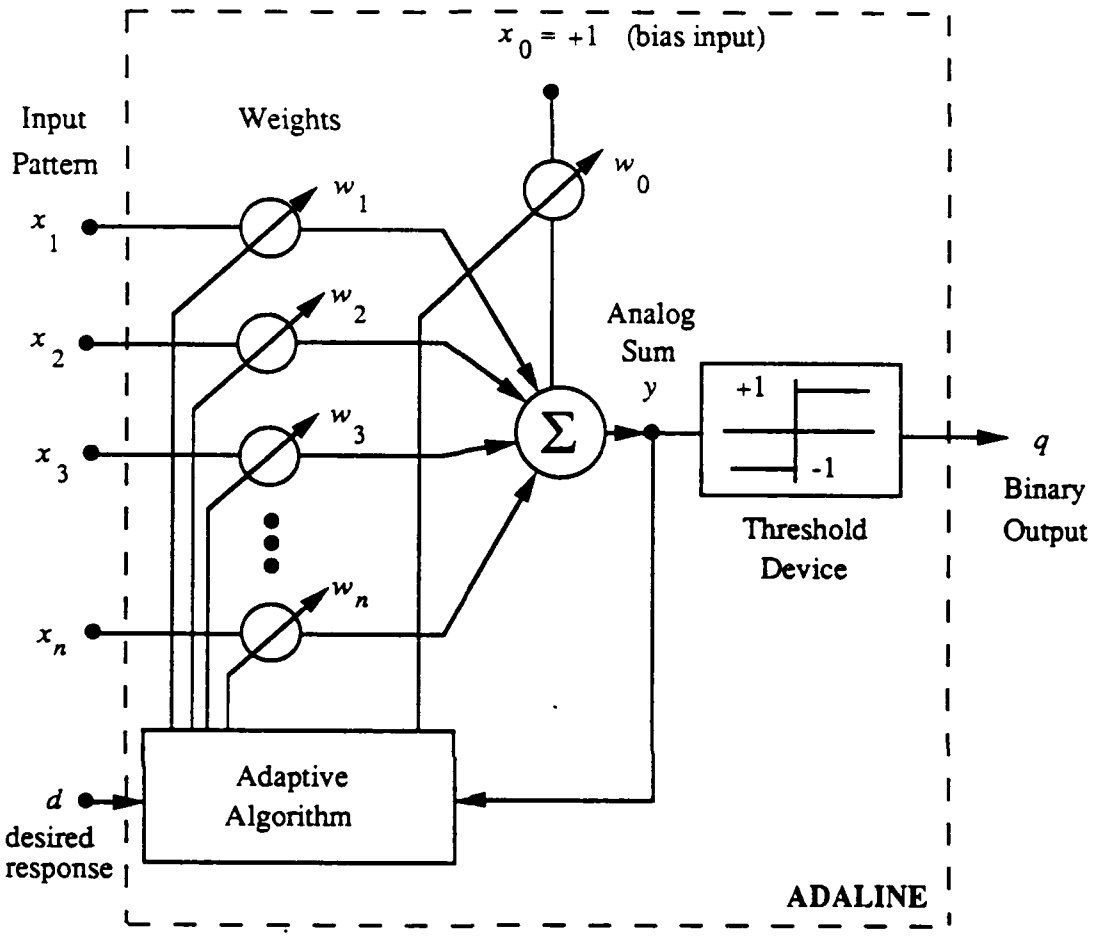


Figure 1.1: The adaptive linear neuron, or Adaline.

1.2 The Adaline

The Adaline is an adaptive threshold logic unit and is depicted in Figure 1.1. The Adaline performs a weighted sum of its inputs and makes a binary decision based on this sum. The Adaline's input pattern has n variable components. The input pattern together with the constant $+1$ bias input form the $(n+1)$-dimensional input vector, $\vec{\mathrm{X}} = [x_0 = +1, x_1, \ldots, x_n]^T$. The input *vector* is the bias augmented input *pattern*, a semantic distinction followed throughout this report. The weighting vector is $\vec{\mathrm{W}} = [w_0, w_1, \ldots, w_n]^T$. The weighted sum, referred to as the Adaline's analog sum or analog output, is the dot product of these two vectors, $y = \vec{\mathrm{X}}^T \vec{\mathrm{W}}$. The Adaline's binary output results from passing the analog sum through a threshold device, $q = \mathrm{sgn}(y)$, where $\mathrm{sgn}(\cdot)$ is $+1$ for positive argument, and

-1 otherwise.

The individual components of the input vector could have analog values but are often restricted to binary values. Except as noted, the input component values will be either $+1$ or -1 in this thesis. There are two reasons for making this restriction. First, the inputs will be binary when an Adaline receives its inputs from the outputs of other Adalines. Second, with this restriction on the inputs, the magnitude of the input vector, $|\vec{X}|$, will be a constant $\sqrt{n+1}$. This simplifies the mathematical analysis presented later.

The weights are allowed to be continuous, real-valued numbers, and are adjusted by an adaptive algorithm. The actual hardware implementation of adjustable, analog-valued weights will not be specifically addressed. The issue was addressed in the past and resulted in the invention of a device called a memistor [4]. Solid state implementations of adjustable weights are the focus of much current research. It is assumed here that such weights are available for the eventual hardware realization of neural networks. The issue of how close these weights have to be to their nominal values will be addressed later.

The Adaline makes binary decisions based upon its inputs and weights. The nature of the Adaline's decision-making capability can be determined by examining its governing equation at the decision threshold, that is, when the analog sum y is zero. The equation that describes the decision separating surface for an Adaline is a dot product relation,

$$y = \vec{X}^T\vec{W} = \vec{W}^T\vec{X} = 0\,.$$

The $\vec{X}$ and $\vec{W}$ vectors can be considered to be located in a continuous, real-valued, $(n+1)$-dimensional vector space. Within this space, the equation above specifies a normality condition. The symmetry of the dot product allows two perspectives in understanding the Adaline.

In the first perspective, consider the weight vector to be fixed in the input vector space. The decision surface is a hyperplane through the origin and perpendicular to the weight vector. It divides the input vector space in half. Those input vectors lying on the same side of the hyperplane as the weight vector form positive dot products with the weight vector and result in $+1$ decisions by the Adaline. Those input vectors on the opposite side of the hyperplane result in -1 decisions. The perspective, then, is that the weight vector defines a division of the input space into decision regions.

The other perspective involves choosing an input vector in the weight space. The input vector has a desired response associated with it. The hyperplane through the origin and perpendicular to the input vector divides the weight space in half. Weight vectors on

only one side of the hyperplane provide the correct desired response for this particular input vector. If now a second input vector is considered, its desired response will also define a half-space where the weight vector must lie to provide a correct response. To accommodate both input vectors, the weight vector must lie in the intersection of the two half-spaces. To satisfy the requirements for an entire training set, the weight vector must lie in the intersection of the half-spaces defined by each input vector and its associated desired response. This intersection is a convex cone emanating from the origin. Any weight vector lying in this cone will be a solution vector for the given training set. The intersection of the half-spaces may be empty. This means there is no weight vector that will provide the desired separation of the input patterns. This second perspective then is that of the input vectors defining a solution region for the weight vector.

The second perspective makes it clear that a single Adaline cannot perform all the possible separations of the inputs. For input patterns having n components, there are 2^n different patterns. A general logic implementation would be capable of classifying each pattern as $+1$ or -1, in accordance with the desired response. Thus, there are 2^{2^n} possible logic functions connecting n inputs to a single output. The single Adaline can realize only a small subset of these functions, known as the linearly separable functions.

When the dimension of the input vectors is small, graphical methods can illustrate the decision-making capability of the Adaline. For the case of an Adaline with two variable inputs, the equation at the decision threshold is:

$$y = \vec{X}^T \vec{W} = w_0 + x_1 w_1 + x_2 w_2 = 0 \,.$$

Rewritten, it becomes:

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2} \,.$$

The plot of the equation is a straight line in the x_1–x_2 input pattern space. The input pattern space is a subspace of the input vector space. It lies in a hyperplane of the input vector space, specifically, the hyperplane defined by $x_0 = +1$. In the case developed here, the input vector and the weight vector are 3-dimensional but the input pattern is 2-dimensional. The decision hyperplane is a 2-dimensional plane and so is the hyperplane containing the pattern space. Both of these planes exist in 3-space, the space in which the input vectors and the weight vector exist. The decision hyperplane intersects the pattern space in a line. In the pattern space, the line has a slope of $-w_1/w_2$ and x_2-intercept of $-w_0/w_2$. A sample plot of this line in the pattern space is shown in Figure 1.2. The line is a decision line. It

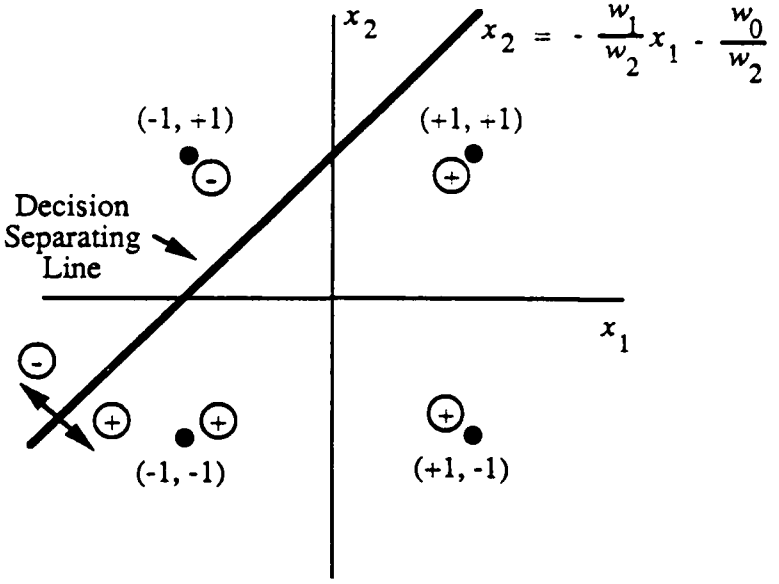


Figure 1.2: Graph of Adaline decision separating line in input 2-space.

divides the input patterns into two sets. Inputs to the right of the separating line cause the Adaline's analog sum to be positive, resulting in +1 decisions by the Adaline. Inputs to the left of the line, result in −1 decisions. The Adaline thus performs a mapping from its two-dimensional input pattern space to its one-dimensional binary output space. The input/output mapping effected in the example of Figure 1.2 is represented by:

$$\begin{aligned}(+1,+1) &\mapsto +1\\ (+1,-1) &\mapsto +1\\ (-1,-1) &\mapsto +1\\ (-1,+1) &\mapsto -1\end{aligned}$$

By adjusting the weights, the slope and intercept of the line in Figure 1.2 can be changed. Reversing the signs of all the weights will give exactly the same separating line but will change which side of the line results in a positive decision. Therefore, changing the weights can cause a change in the input/output mapping of the Adaline. Methods for adapting the weights of a single Adaline to achieve a desired input/output mapping were developed in the early 1960's.

The mechanism for changing the weights is represented by the "adaptive algorithm" block in Figure 1.1. The algorithm's inputs are the desired response, *d*, for the pattern being presented and the actual analog response of the Adaline. Mays [5] presents three adaptation procedures for the single Adaline. All three of these procedures produce a set

of weights that provide the desired input/output mapping, if such weights exist, in a finite number of adaptations. An example of an input/output mapping that cannot be achieved by a single Adaline will be shown later.

Mays' formulation of the adaptation procedures includes a concept called the deadzone. In any hardware implementation of an Adaline, a real thresholding element will be used. If the analog sum y is very close to zero, it is possible that the thresholding element could change output states in an erratic fashion due to noise. Manufacturing tolerances might also cause the actual threshold to be different from zero. For these reasons it is desired that the magnitude of the analog sum be greater than a positive deadzone value δ. The magnitude of the analog sum is called the "confidence level." The confidence level is a measure of how sure the Adaline is about its decision. During training then, not only must the binary decision be correct, but the confidence level must be greater than the deadzone value.

All of the Adaline adaptation procedures can be summarized as follows. There exists a set of patterns and associated desired responses that the Adaline is to learn. This set is called the training set. Present a pattern and its associated desired response from the training set to the Adaline. If the binary response is correct and the confidence level is greater than the deadzone value, go on to the next pattern. If the response is incorrect or the confidence level is less than the deadzone value, make a change in the weights. The weights are changed by adding or subtracting some portion of the input. The weight changing method that Mays called the modified relaxation procedure is detailed as follows:

$$\begin{aligned} \vec{W}(k+1) &= \vec{W}(k) && \text{for } d(k)\vec{X}^T(k)\vec{W}(k) \geq \delta \\ &= \vec{W}(k) + \frac{\eta d(k)}{n+1}\vec{X}(k)[L - d(k)\vec{X}^T(k)\vec{W}(k)] && \text{for } d(k)\vec{X}^T(k)\vec{W}(k) < \delta \quad (1.1) \end{aligned}$$

Here, k is an iteration cycle number so that $\vec{W}(k+1)$ is the weight vector after the kth adaptation. As before, $d(k)$ is the desired binary response, $+1$ or -1, associated with the input vector for the kth iteration, $\vec{X}(k)$. The adaptation constant, η, must have value $0 < \eta \leq 2$ to insure convergence. The deadzone can be selected $0 < \delta < 1$. The quantity L is called the adaptation level and is selected such that $\delta < L$. Generally, L is selected to be $+1$. The adaptation level can be thought of as a target confidence level when $\eta = 1$. To see this, compute the resulting analog sum after adaptation by premultiplying both sides of Equation 1.1 by $\vec{X}^T(k)$, remembering that $y(k) = \vec{X}^T(k)\vec{W}(k)$, and $\vec{X}^T(k)\vec{X}(k) = n+1$.

$$y(k+1) = y(k) + \eta d(k)[L - d(k)y(k)]$$

$$= y(k)[1-\eta] + \eta d(k) L$$

$$= d(k) L \qquad \text{for } \eta = 1.$$

After adaptation, the Adaline will provide the correct response with confidence equal to the adaptation level when $\eta = 1$ is used in the modified relaxation scheme.

1.3 Early Madalines

As noted before, there are some input/output mappings that a single Adaline cannot realize. The single Adaline can only realize the linearly separable functions. From Figure 1.2 it can be seen that inputs requiring plus decisions must be separable from those requiring a minus decision by a straight line. In higher dimensional input spaces, this requirement generalizes to requiring the separation be done by a hyperplane. An input/output mapping that is not linearly separable for the two variable input case is the exclusive-or function represented by:

$$\begin{aligned} (+1,+1) &\mapsto -1 \\ (-1,+1) &\mapsto +1 \\ (-1,-1) &\mapsto -1 \\ (+1,-1) &\mapsto +1 \end{aligned} \tag{1.2}$$

The number of binary functions of n variable inputs is 2^{2^n}. For $n = 2$, 14 of the 16 functions are linearly separable. While no general formula exists for determining how many of the possible functions of n variables are linearly separable for general n, it is known that the fraction becomes very small for even moderate values of n. For example, at $n = 5$ only 94,572 of the possible 4.3 x 10^9 binary functions are linearly separable [6]. Thus, a single Adaline's ability to realize arbitrary input/output mappings in high dimensional input spaces is limited.

To combat this limitation of the Adaline, Ridgway [6] used simple networks of Adalines which were called Madalines. The form of Madaline investigated by Ridgway is shown in Figure 1.3. It consists of a layer of Adalines whose outputs feed into a fixed logic element. All of the Adalines receive the input vector $\vec{X}$ as an input. The Madaline's response is taken at the output of the fixed logic unit and is compared with the desired response associated with a particular $\vec{X}$ during training.

Some of the fixed logic units used by Ridgway implement the AND, OR and majority functions. All of these logic units can be realized by an Adaline with fixed weights as shown

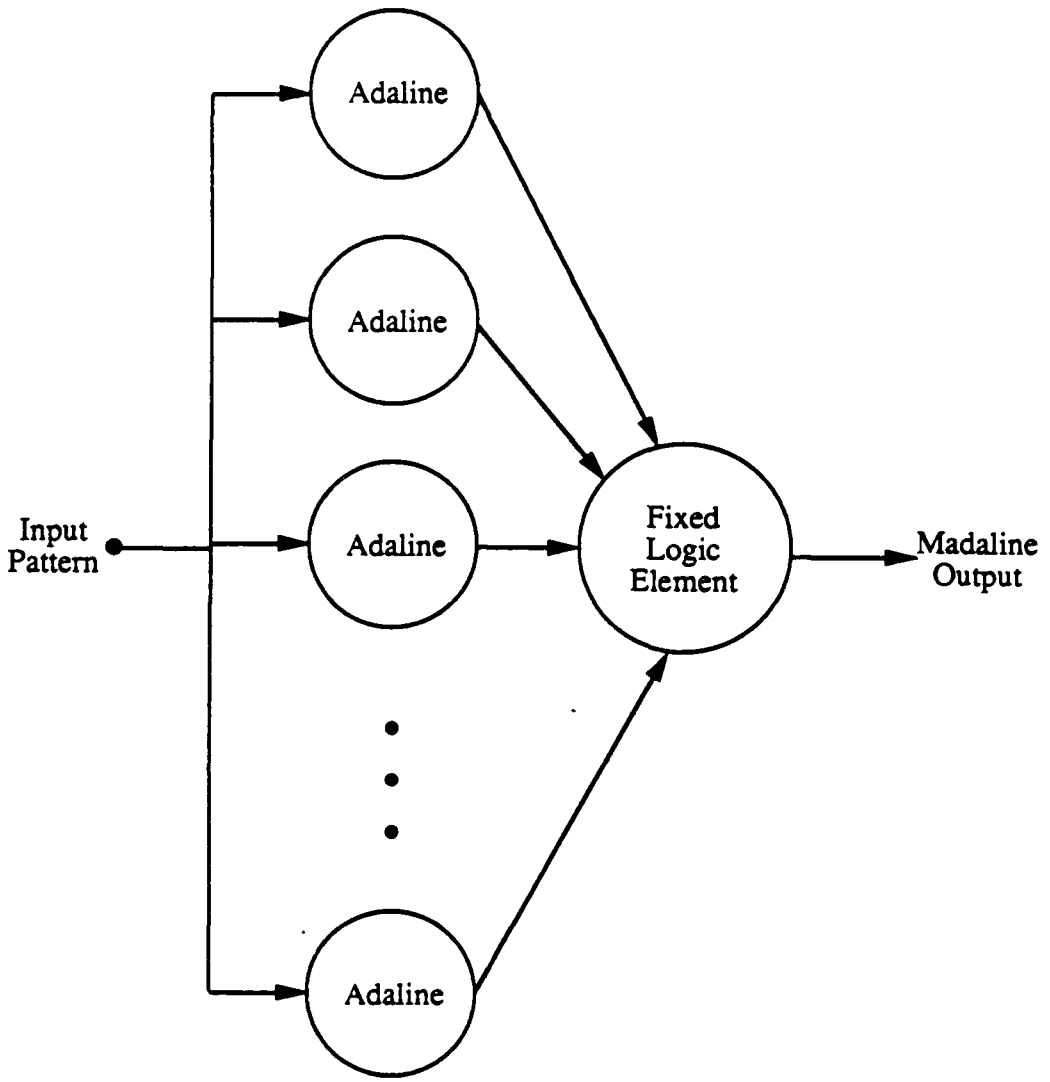


Figure 1.3: Structure of the Ridgway Madaline.

in Figure 1.4. The weights shown in this figure are not unique but do realize the required logic function. Thus, Ridgway's Madalines can be thought of as the simplest of 2-layer feed-forward Adaline networks, the second layer being restricted to a single nonadaptive Adaline.

Ridgway presented an algorithm to train Madalines of the type in Figure 1.3. (A historical note: The algorithm was actually developed by Widrow and Hoff, but first published by Ridgway [6].) Because the logic element is fixed, it is possible to determine which

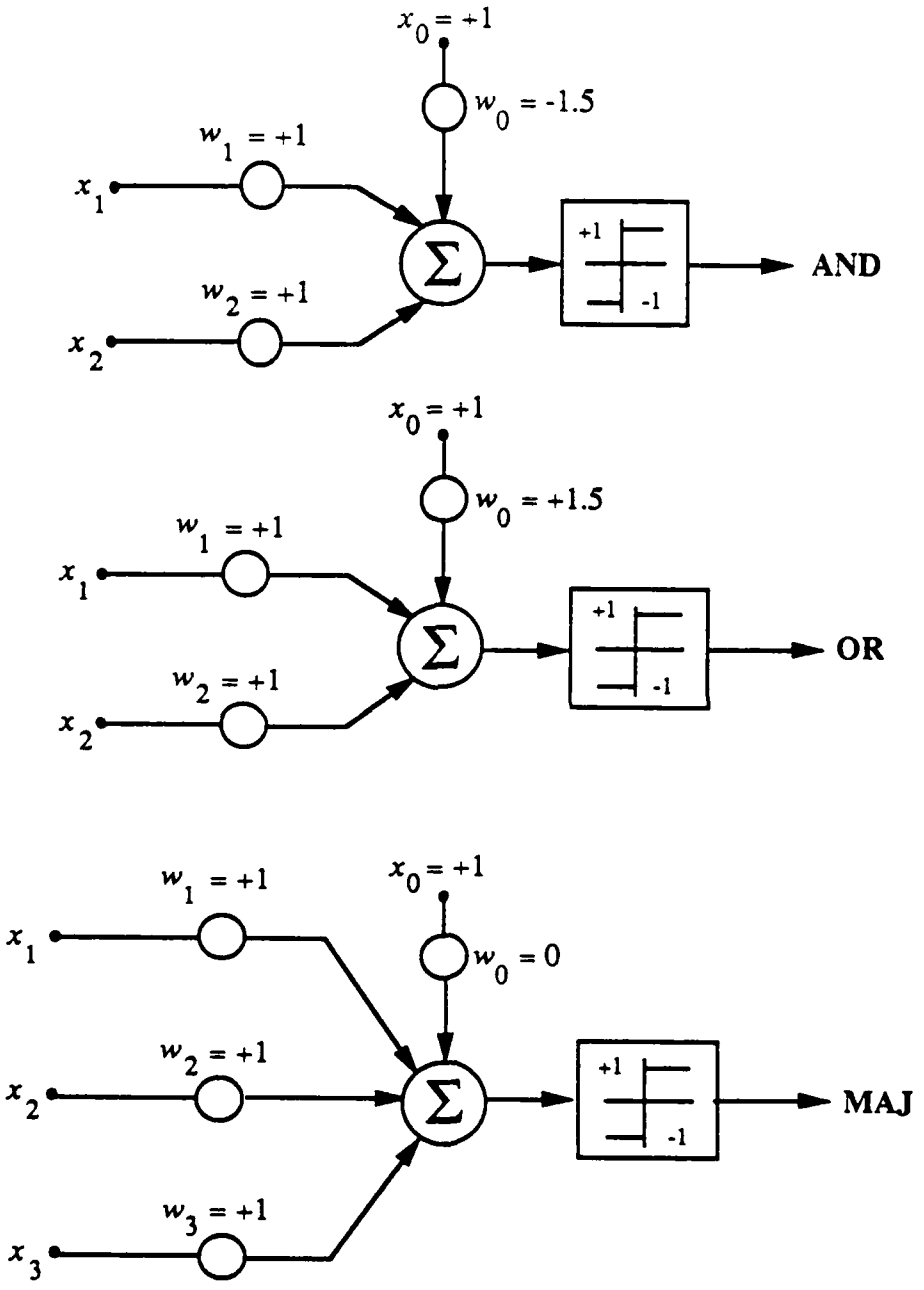


Figure 1.4: Implementation of the AND, OR and MAJority functions using Adalines with fixed weights.

Adaline(s) are contributing to any output errors that occur during training. The determination of which elements in a network contribute correctly to the network's overall output is commonly referred to as the credit assignment problem.

Consider the case where the logic element is an OR. A multi-input OR element makes a $+1$ decision whenever one or more of its inputs is $+1$. It makes a -1 decision only when all of its inputs are -1. Suppose a pattern is presented and the desired response is -1 but the Madaline's actual response is $+1$. To correct the response, all those Adalines with $+1$ outputs need to be adapted to provide minus responses. These adaptations are done using the methods outlined by Mays. Suppose instead the desired response is $+1$ but the actual response is -1. This means all of the Adalines are responding with -1. One or more of the Adalines need to be adapted to respond with $+1$ to correct the Madaline's overall response. The rule that Ridgway presented adapts only one Adaline and chooses the one with the lowest confidence, that is, the one whose analog sum is closest to zero. Of course, if the actual response is correct, no changes are made.

The algorithm for training the Madaline of Figure 1.3 will be called Madaline Rule I throughout this paper. It can be summarized as follows:

- Present a pattern to the Madaline. If the Madaline output and the desired response match, go on to the next pattern. Make no adaptations.

- If an error occurs, use knowledge about the fixed logic device to determine which Adaline(s) are contributing to the erroneous output. Select a minimum number of Adalines such that if their outputs reverse state, the Madaline will respond correctly. If this minimum number does not include all of those contributing to the error condition, select those with the lowest confidence levels to be adapted. Use one of Mays' procedures to adapt the selected Adaline(s) in such a way as to reverse their outputs. (Note: Mays' procedures will not necessarily cause an Adaline to reverse state. The Adaline's analog response will be changed in the direction to provide the new response. Depending upon the adaptation constant, this change may not be large enough to actually change the Adaline's binary response.) Go on to the next pattern.

- Repeat this procedure until all training patterns produce the correct response.

Ridgway also points out that the pattern presentation sequence should be random. He found that cyclic presentation of the patterns could lead to cycles of adaptation. These cycles would cause the weights of the entire Madaline to cycle, preventing convergence.

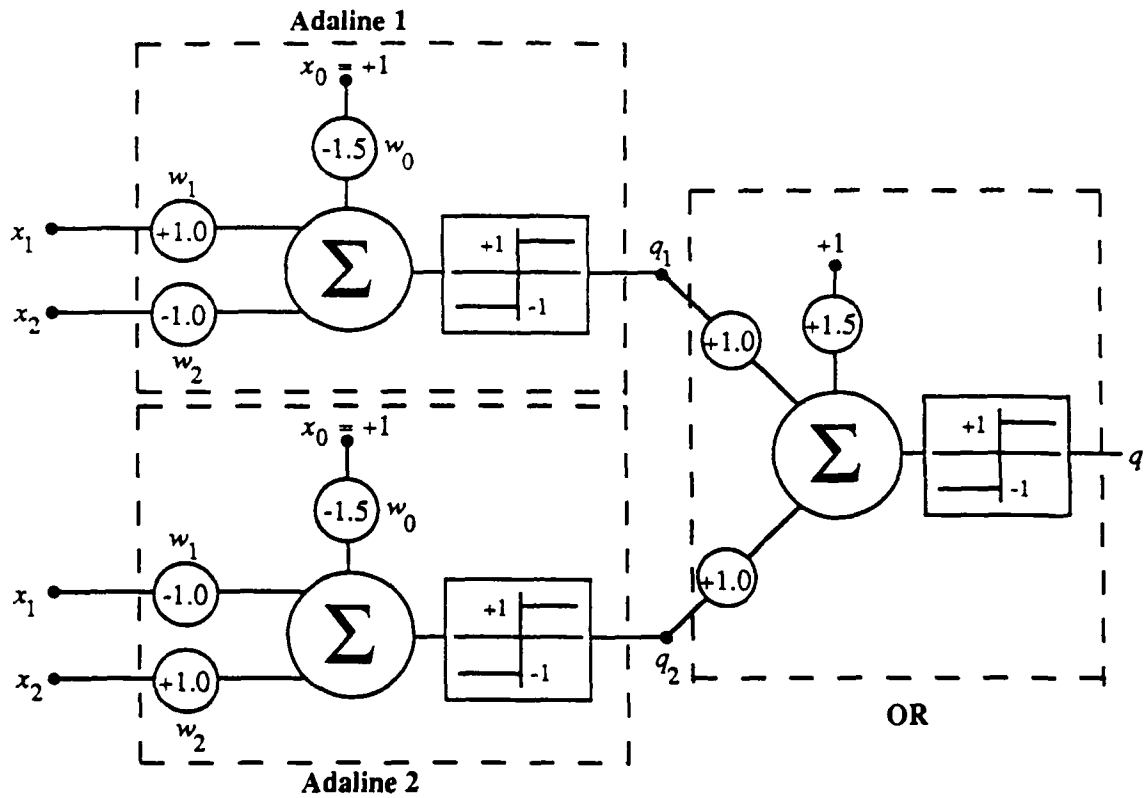


Figure 1.5: Ridgway Madaline realization of the exclusive-or function.

Adaptations are being performed to correct the weights for the pattern being presented at that time. It is not obvious that these weight changes will contribute correctly to a global solution for the entire training set. Ridgway presents an argument for convergence of the procedure. His argument is of a probabilistic nature. He shows that good corrections will outnumber bad corrections on the average. Thus, a global solution will be reached after enough time with probability one.

The exclusive-or problem represented in Equation 1.2 can be solved by the Ridgway Madaline shown in Figure 1.5. Here, the fixed logic element is the OR. Figure 1.6 is a graphical depiction of how this network works. The outputs of the two Adalines in Figure 1.5 are labeled q_1 and q_2. The Adalines map the input pattern $x_1 - x_2$ space to an intermediate $q_1 - q_2$ space that the OR element can separate as required. This mapping of the input space to an intermediate space that is then linearly separable is the essence of how layered Adaline networks operate.

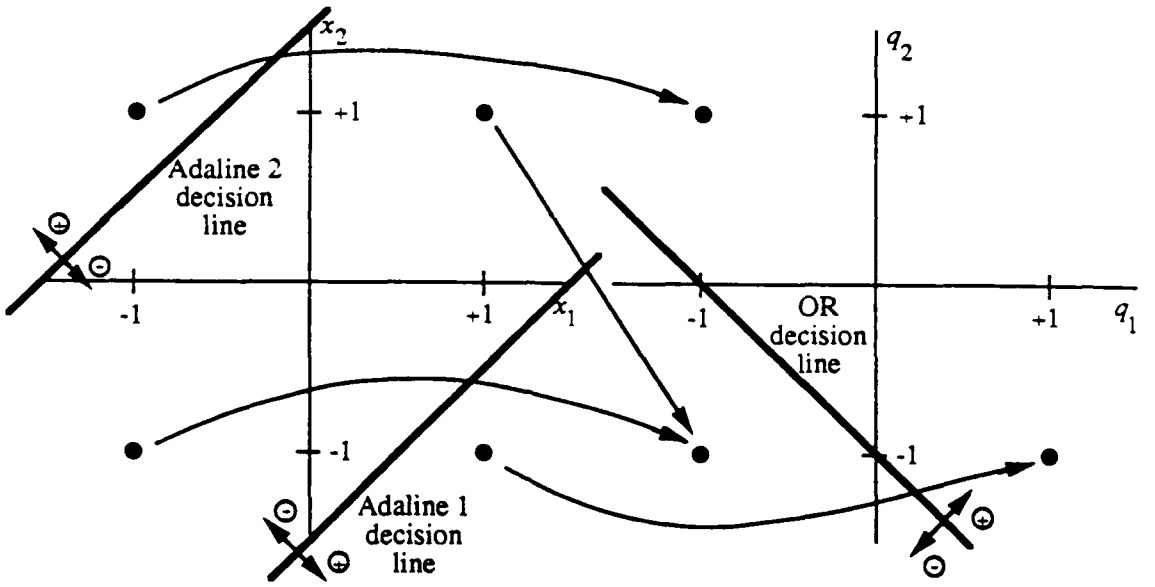


Figure 1.6: Graphical presentation of the Madaline of Figure 1.5. The Adalines map the input space into an intermediate space separable by the OR unit.

1.4 Concept of Madaline Rule II — Minimal Disturbance

Madaline Rule II is a training algorithm for Madalines more complicated than those of Ridgway's. The general Madaline has multiple layers. Each layer has an arbitrary number of adaptive Adalines. Figure 1.7 shows an example of a 3-layer Madaline. This work presents experimental results of training 2-layer Madalines with MRII. The procedure generalizes to Madalines having more than two layers of Adalines, though no results of training such Madalines are presented.

Figure 1.7 introduces some terminology to describe the general Madaline. Let l be the number of layers in the network. In the case of Figure 1.7, $l = 3$. To remain consistent with previous notation, the input vector $\vec{X}$ has n variable components plus a constant bias component and is the input to all the first-layer Adalines. Since the outputs of the first-layer Adalines are the variable inputs of the second-layer Adalines, let n_1 be the number of Adalines in the first layer. The outputs of the first-layer Adalines represent an intermediate pattern in the input/output pattern mapping scheme done by the Madaline. The current literature often refers to the pattern as being "hidden." Therefore, let $\vec{H}^1 = [h_0^1 = +1, h_1^1, \ldots, h_{n_1}^1]^T$ be the input vector to the second-layer Adalines. The constant bias input to the second layer is not pictured and is assumed to be provided internally to the Adalines as in the general

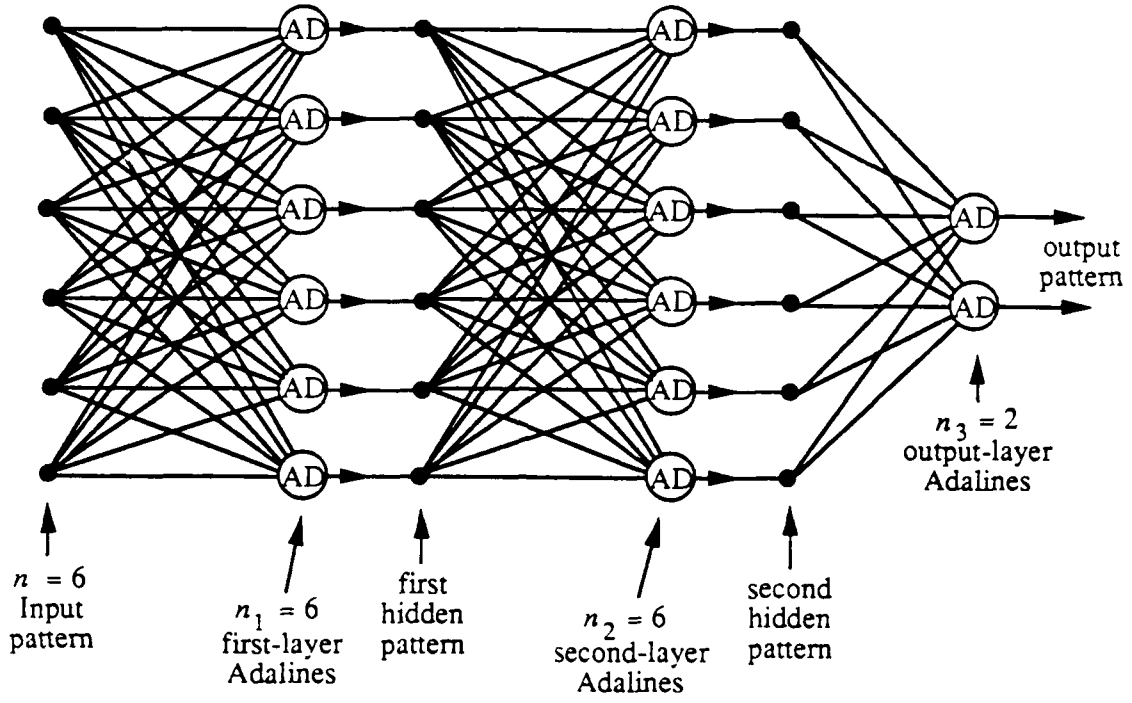


Figure 1.7: A three layer example of the general Madaline

Adaline of Figure 1.1. Similarly, $\vec{H}^2$ with n_2 variable inputs plus a constant bias input is the input vector to the third layer. The final layer of Adalines is called the output layer since the response of these Adalines constitute the Madaline's response. The desired and actual responses are vectors. The actual output vector is $\vec{O} = [o_1, \ldots, o_{n_l}]^T$. The desired response vector is designated $\vec{D}$. A single component of these vectors is called a bit, being the actual or desired binary response of a particular output-layer Adaline. The configuration of a particular Madaline is referred to as an n_1-feed-n_2-...-feed-n_l network with n inputs. Thus, the net in Figure 1.7 is called a 6-feed-6-feed-2 Madaline with 6 inputs.

At this point, it should be noted that the layered feed-forward Madaline is not the only structure being investigated by researchers in the neural network field. A layered feed-forward network which has units that are similar to the Adaline is used by Rumelhart, et al. [7]. Instead of using a threshold device, Rumelhart's units use a "sigmoid" function to operate on the weighted sum of the inputs. The sigmoid function is continuous and differentiable, unlike the "sign" function of the threshold device. Networks composed of Adaline elements are used by Hopfield [8] and Hinton, et al. [9]. The Hopfield network and Hinton's Boltzmann machine use a fully connected architecture, an architecture in

which every unit has an input from the output of every other unit, instead of the layered feed-forward architecture of the general Madaline.

The credit assignment task for the general Madaline is much more complicated than that for the Ridgway Madaline. Suppose a Madaline has three output Adalines and for a particular input, only one of the output Adalines' responses agrees with the corresponding desired response. How much is the output of any particular Adaline in the first layer helping or hurting the overall response of the network? One way to check is to reverse the output of the Adaline and let this change propagate through the network. In some instances, the Madaline's output may not change at all. Sometimes the number of output errors will increase, and sometimes they will decrease. At other times, outputs will change but no net gain or loss of correct responses will occur. Faced with all these possibilities, it is difficult to know which Adalines in a network need to be changed when errors occur during training. The adaptation techniques for the single Adaline and the Ridgway Madaline provide some insight for developing a method, though.

All of the algorithms examined so far make no changes to the network if the current response matches the desired response (assuming no deadzone criterion is being used). When changes are made in the Ridgway Madaline, the fewest number of Adalines are changed and those of lowest confidence are selected. This is because the weights of the low confidence Adalines need to be changed least to effect a change of the output. To see this, suppose a generic weight update rule is used:

$$\vec{W}(k+1) = \vec{W}(k) + \Delta\vec{W}(k)$$

Premultiplying both sides by $\vec{X}^T(k)$,

$$y(k+1) = y(k) + \vec{X}^T(k)\,\Delta\vec{W}(k)$$

so that,

$$\begin{aligned} \Delta y &= y(k+1) - y(k) \\ &= \vec{X}^T(k)\,\Delta\vec{W}(k) \\ &= |\vec{X}(k)||\Delta\vec{W}(k)|\cos\theta \end{aligned}$$

where θ is the positive angle between $\vec{X}(k)$ and $\Delta\vec{W}(k)$. The change in the Adaline's analog response is greatest for a given magnitude of weight change when $\Delta\vec{W}(k)$ is selected aligned

with $\vec{X}(k)$. This is the type of weight update correction used by Mays' methods as well as the LMS least means squares algorithm [1, 10]. (Mays' procedures are actually generalizations of the LMS algorithm.) Thus, the Adaline and Madaline procedures seen thus far allow the current pattern presented to the system to be trained in with least overall disturbance to the system. This is important because adaptations are made based only upon the current input. By changing the overall system as little as possible, there is less likelihood of disturbing the responses for other previously trained in patterns from the training set.

The procedure of making corrections only when errors occur, and then making corrections that are least disturbing to the overall system is called the minimal disturbance principle. This idea can be used to formulate a strategy for training the general multi-layer, multi-output Madaline.

The procedure is as follows. Present a pattern from the training set to the network. Count the number of output Adaline responses that do not match their respective desired responses. This number is the Hamming distance between the desired response vector and the actual output vector. Select the least confident Adaline in the first layer. Perform a trial adaptation by reversing the response of this Adaline, that is, if the selected Adaline had been responding $+1$, cause it to respond -1 or vice versa. This change will propagate through the network and perhaps change the output Adalines' responses. If the Hamming distance between the new output response and the desired response is reduced, accept the reverse adaptation. If the number of errors is not reduced, return the reverse-adapted Adaline to its previous state. If any errors remain at the output, perform a reverse adaptation on the next least confident Adaline of the first layer. Again, accept or reject this reverse adaptation depending upon whether the number of output errors is reduced. Continue in this fashion until all output Adalines respond correctly, or all single Adaline reverse adaptations have been made. If errors remain at this point, proceed by reverse adapting the first-layer Adalines two at a time, beginning with the pair which is least confident. The criterion for acceptance of the trial is the same as before. If pairwise trials are exhausted, one can use reverse adaptations involving three Adalines at a time, then four at a time, etc., until the output errors are zero. If all possible trial adaptations are exhausted without reducing the errors to zero, repeat the procedure using the Adalines of the second layer, then the third layer, etc., until finally reaching the output layer. The output layer, of course, can be corrected to give the desired outputs by adapting each erroneous Adaline to give the correct response. If the output layer Adalines were adapted at the beginning of the procedure, their outputs would have been corrected immediately. This would not be useful,

however, since the hidden layers would never adapt. The output layer would be doing all of the "adaptive work."

The basic philosophy is to give responsibility for corrections to those Adalines that can most easily assume it, that is, make the response for the current input correct with the least overall change to the weights in the network. The details of how to perform trial adaptations, how confident an Adaline should be after it is adapted, how to choose the ordering of possible pairwise trial adaptations, etc. are deferred to Chapter 2, and are the author's specific contributions.

Chapter 2

Details of Madaline Rule II

This chapter presents the actual details of how MRII works. The reader will be able to write computer simulation code after reading the chapter.

The chapter begins by discussing some of the features desired of a neural network training algorithm. These desirable features require modifying the minimal disturbance principle in order to develop a practical implementation. The MRII algorithm was developed empirically. As the algorithm evolved, it was necessary to introduce a concept called "usage." This concept further modifies the minimal disturbance principle. The usage concept, why it is needed, and its implementation are discussed.

2.1 Desired Features for a Training Algorithm

A neural network is a collection of relatively simple processing elements. The elements are connected together in such a way that the collection exhibits computational capabilities way beyond those of a single element. The specific computation the network performs is "trained in" by presenting a collection of sample inputs and desired responses to the network. It is reasonable to assume that the network can be trained off-line, that is, the training data is available for presentation as many times as necessary via some external storage and presentation system. It will be assumed that training can take a relatively long but not unreasonable amount of time. Once trained, the neural network's utility is realized by its ability to respond almost instantaneously to new inputs presented to it. This speed is due to the fact that the computation is distributed among all the processing elements. In a layered feed-forward Madaline, the response time will be roughly the number

of layers multiplied by the time required for a single Adaline to perform its weighted sum and thresholding operation.

The hardware implementation of neural networks must contend with the problem of connectivity. The neural net relies on a high degree of connectivity to perform. The realization of the high fan-in and fan-out needed for neural networks is a definite hardware challenge. The issue to be made here is that the training algorithm must not exacerbate this problem. The training algorithm must be implemented with a minimum of added connections.

For MRII, there is a need for a master controller to direct the trial adaptations and decide which are accepted and rejected. Communications between network components require hardware and time, two quantities to be minimized in any implementation of the algorithm. Therefore, the controller should operate with a minimum amount of communication. Information about a specific Adaline's weight values or analog sum should not be needed by other units or the master controller.

While it has been assumed that training time is not a prime consideration, this time has to be reasonable. The question of how training time grows with network size cannot be completely ignored. Any algorithm implementation that requires an exponential increase of training time as the size of the network increases will not be acceptable.

The next section shows how to implement the concepts of minimal disturbance with the above considerations in mind.

2.2 Implementing MRII

This section presents a block diagram of the implementation of MRII. The purpose here is to present the implementation at a high level of abstraction, not to present a wiring diagram. The function of each block in the diagram is explained. Its hardware realization is not addressed.

The basic methodology of MRII was presented in Chapter 1. Its implementation requires several things. The first is the ability to perform trial adaptations by layers. The first layer is trial adapted first and the output layer adapted only when all of the output errors are not corrected by the previous layers. Within a layer, there must be the ability to involve different numbers of Adalines in the trials. Trials involving only a single Adaline will be done first. Then, if necessary, pairwise trials, that is, trials involving two Adalines at a time, will be done. These trials will be followed by threewise, fourwise, etc., trials. Finally,

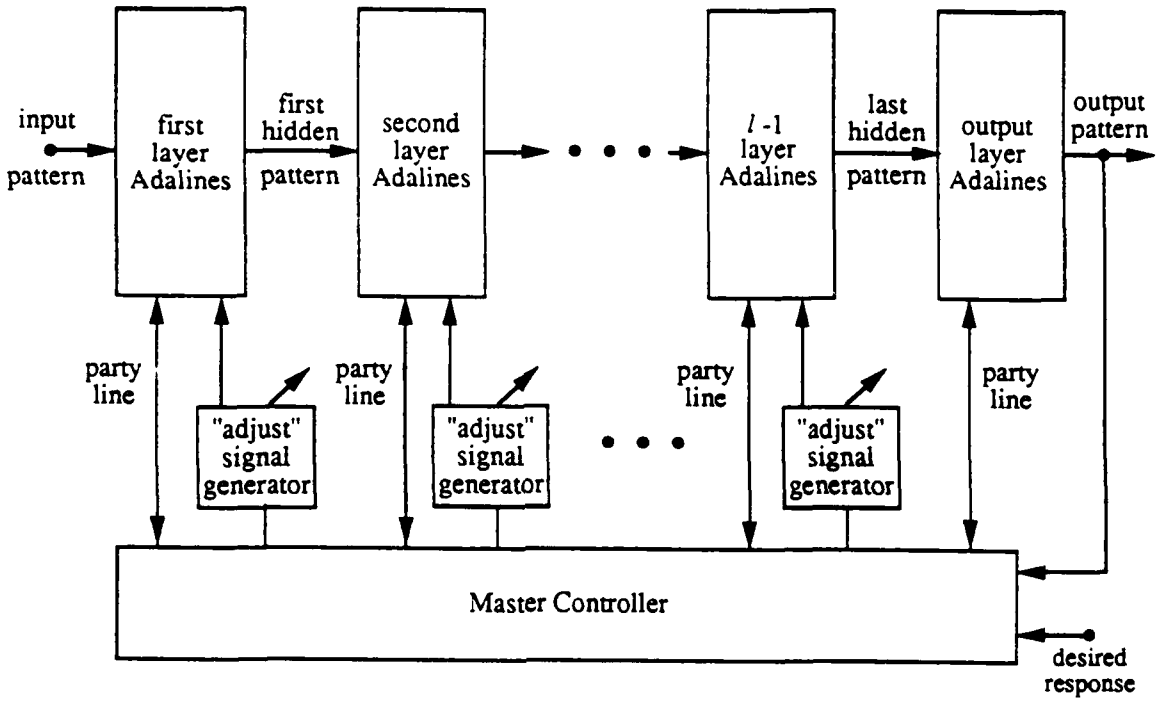


Figure 2.1: Block diagram implementation of a Madaline trained by MRII.

it is necessary to select Adalines for trial adaptations in order of increasing confidence. Those Adalines with analog responses closest to zero are to be trial adapted before the more confident ones.

Figure 2.1 shows a Madaline at the highest level of abstraction. There is a master controller that communicates with each layer of Adalines in the network by means of a two-way "party line." The master controller also controls an "adjust" signal generator for each layer of the Madaline except the output layer. Figure 2.2 shows the structure of a single layer within the network. The party line and adjust signal line are connected in parallel to each Adaline of the layer.

The party line is the communications link for command and control during trial reverse adaptations. Communication on this party line is structured so that only one element is transmitting at a time. An important aspect of this party line is that if an Adaline transmits on it, all other Adalines receive this transmission, not just the master controller. This is why it is called a party line.

The minimal disturbance principle requires that trial adaptations begin with the least confident Adalines. This implies that the Adalines on a particular layer will need to be

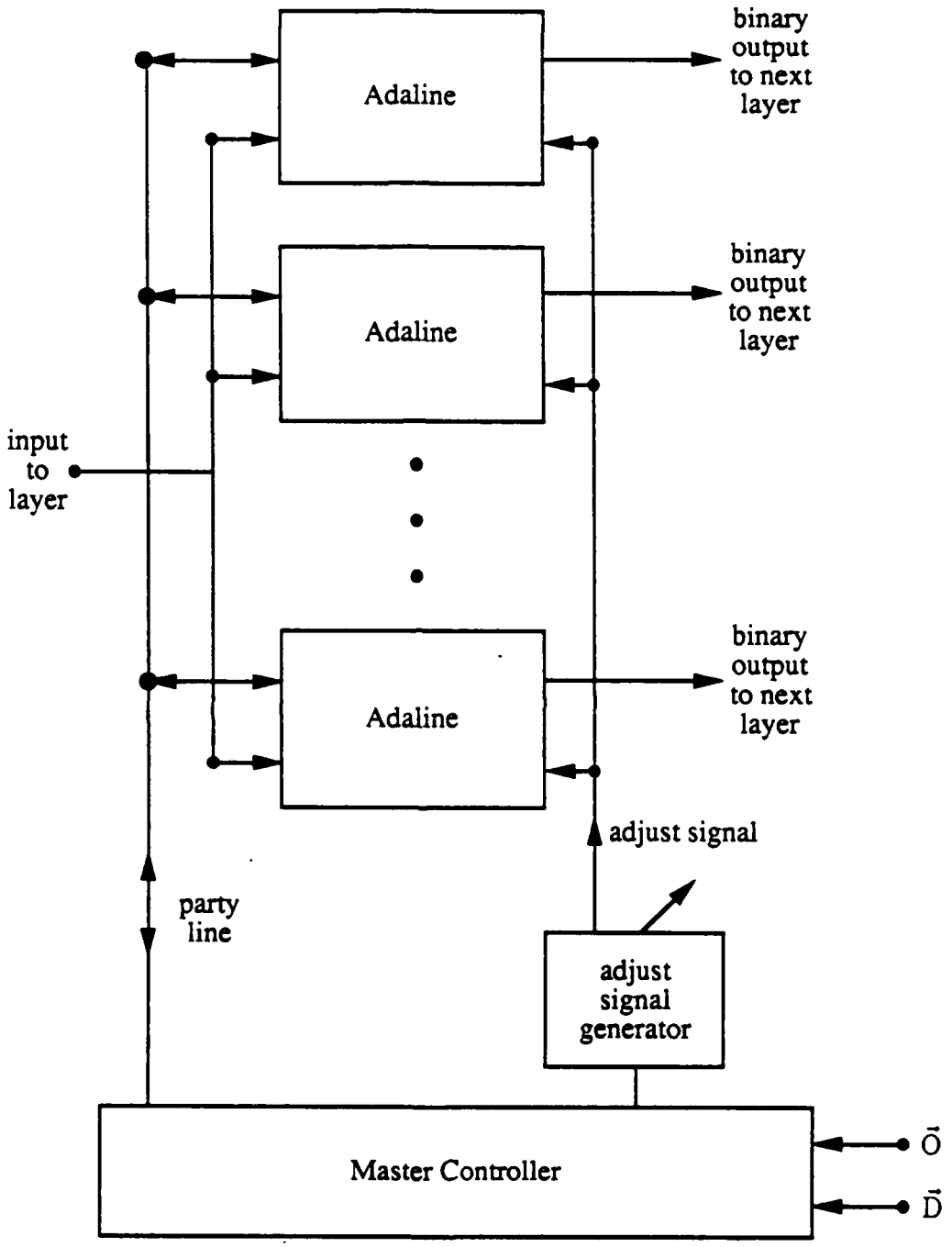


Figure 2.2: Structure of a single layer within the network of Figure 2.1.

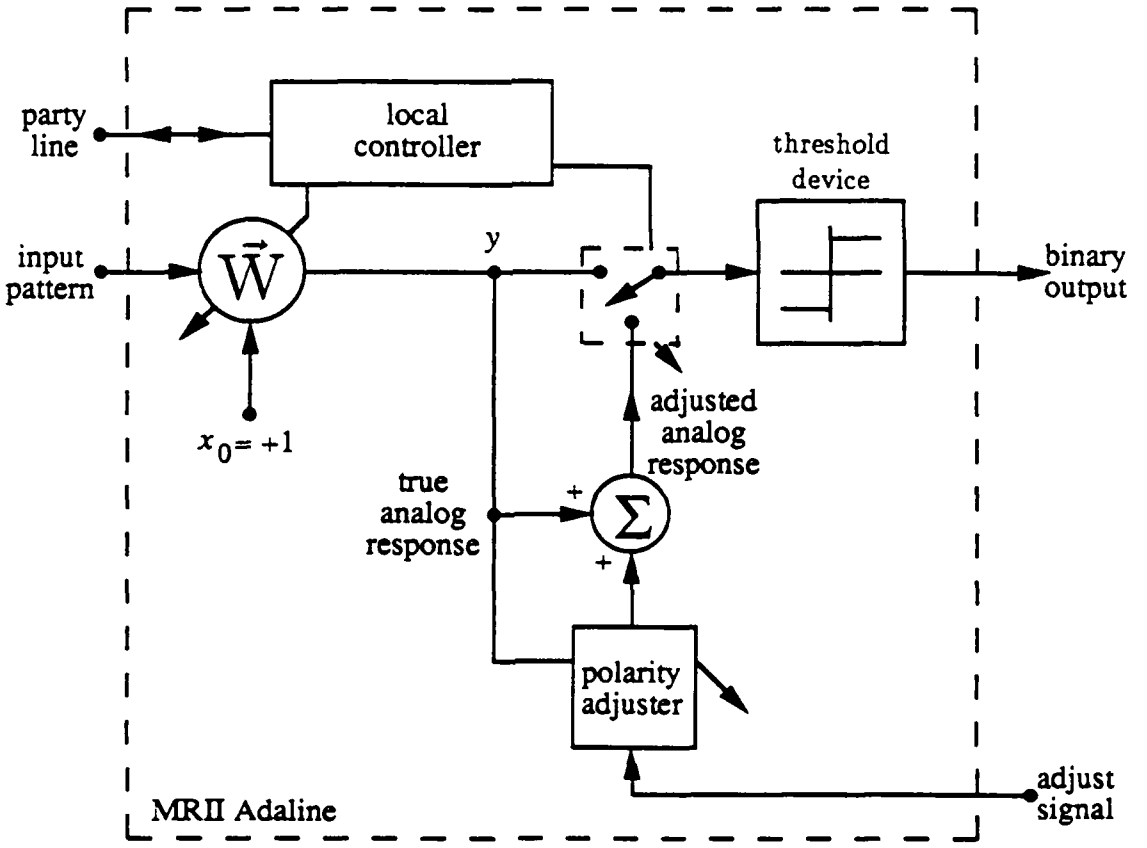


Figure 2.3: Block diagram of the Adalines for Figure 2.2.

sorted by the value of their analog response. To do this without actually communicating analog values around the network is the purpose of the adjust signal.

Figure 2.3 shows a block diagram of the control for one Adaline for an adaptive layer as in Figure 2.2. The Adaline uses the adjust signal to modify its analog response during trial adaptations. The Adaline's local controller operates an internal switch. This switch selects either the true analog response or an adjusted analog response as input to the threshold device. The adjusted analog response is formed by adding the true analog response to the output of the polarity adjuster. The output of the polarity adjuster is the adjust signal coming from the master controller with polarity opposite to that of the true analog response.

When the master controller wants to trial adapt a layer, it transmits on the layer's party line that the layer is to begin trial adaptation. The local controller of each Adaline on the adapting layer throws its internal switch so that its output is modified by the adjust signal. The adjust signal is initially zero. The master controller then slowly increases the adjust

signal. Each Adaline on the layer being adapted adds the adjust signal to its true analog response with proper polarity to reduce the apparent confidence of the Adaline. As the adjust signal increases, the adjusted analog response eventually crosses zero and takes on a sign opposite to that of the true analog response. The binary response reverses and the Adaline is said to be "trial-adapted." As the adjust signal increases, the reversal of outputs will occur in order from least confident Adaline to most confident Adaline. At the time of reversal, the trial-adapted Adaline transmits on the party line that it has reversed.

If trials involving one Adaline at a time are being performed, the master controller stops increasing the adjust signal when the first Adaline reverses. It waits a sufficient time for changes to propagate through the net, and then checks the actual output response versus the desired response for improvement. If improvement occurs, the master controller will transmit an "adapt" command. The trial-adapted Adaline changes its weights to provide the new binary response. The weight change is made using the modified relaxation rule. The parameters for the weight update are detailed later but are chosen such that the weights change sufficiently to actually reverse the Adaline's output. If the Hamming distance between the desired and actual output responses did not decrease, the master controller transmits a "reset" command. The trial-adapted Adaline throws its internal switch to disconnect its output from influence by the adjust signal. The Madaline returns to its original state before trial adaptation. The master controller then increases the adjust signal until the next least confident Adaline reverses its output, etc.

If trials involving two Adalines at a time are being performed, the master controller increases the adjust signal until two Adalines transmit that they have reversed their outputs. The master controller then checks the Madaline's output performance. Again, if an improvement occurred, the master controller transmits the adapt command. Now, both of the Adalines participating in the trial change their weights. If output performance did not improve, the master controller transmits reset. Now, only the first Adaline of the trial-adapted pair disconnects from the adjust signal. This is why the Adalines need to be able to hear each other on the party line. They have to be able to keep track of their position in the sequence of output reversals. Only in this way can they know when a reset command applies to them. After the first Adaline resets, the master controller increases the adjust signal until another Adaline reverses its output. At this point, the pair of Adalines participating in the trial adaptation are the second and third least confident of those when the trial sequence began. If no improvement in output performance is obtained, the reset command is given and the second least confident Adaline disconnects from the adjust signal.

The adjust signal then increases until the third and fourth least confident Adalines are trial adapted, etc.

Three at a time trial adaptations are handled similarly. The adjust signal is increased until three Adalines reverse output. Then the output performance is checked. If the trial is rejected, the first Adaline of the three that reversed outputs disconnects from the adjust signal.

Any time a trial adaptation is accepted, the master controller restarts the adaptation procedure with trials involving one Adaline at a time. This is done because after an Adaline adapts its weights, its new analog response will cause it to have a different confidence than before. Also, since the response of the layer has changed, some of the trial adaptations previously rejected may now be accepted. This causes what really is an accepted pairwise adaptation to masquerade as two accepted single adaptations. In similar fashion, a three at a time adaptation may be accepted as a single and a pair. Experience simulating the algorithm shows that it is rarely necessary to trial adapt more than three Adalines at a time when the layer has less than twenty Adalines. The number of accepted single trial adaptations outnumbers the accepted pairwise trials by a factor of ten. The accepted threewise trials are about one-fourth the number of pairwise acceptances. Due to this diminishing acceptance rate, performing trials involving more than three Adalines at a time is not worth the time it takes to do them.

With this implementation, the least confident and third least confident Adalines will not be considered as a pair during pairwise trial adaptations. Thus, some good trial adaptation possibilities are not be considered. The sum of the confidences of the first and third least confident Adalines will be less than that of the second and third least confident. Based on minimal disturbance only, the trial adaptation involving the first and third least confident Adalines should be considered first. The implementation modifies the minimal disturbance principle to insure a practical implementation. This scheme insures the training time will not grow at a exponential rate as the number of Adalines in a layer increases. Training time would grow as such if all possible combinations of single, pairwise, threewise, fourwise, etc., trial adaptations were considered. A more complicated implementation would also be needed to consider these other trials.

As mentioned earlier, the weights of an Adaline are adapted using the modified relaxation method of Equation 1.1. It was shown in Chapter 1 that if η is chosen as 1, the confidence of the Adaline after adaptation will equal the adaptation level, L, and its binary output will be its new desired response. The new desired response for the Adaline being

adapted is either +1 or −1, the sign being the same as the current trial output of the Adaline. Experience indicates that $\eta = 1$ and L = .2 are good parameters to use. Setting $\eta = 1$ insures the Adaline will be responding with the new output value after the weights are changed. Any value for η that is less than 1 will not guarantee the reversal of the output. For MRII, it is essential that the weights change enough to actually reverse the outputs of those Adalines participating in an accepted trial adaptation. Unlike the procedures for adapting the single Adaline and the Ridgway Madaline, MRII is not a "one-shot" procedure. Several trial adaptations may need to be accepted before all of the output errors are corrected. When a trial is accepted, the output reversals of the participating Adalines must be realized by weight changes before further trials are made. At the same time, the changes in the weights should be small in keeping with the minimal disturbance principle. Given that $\eta = 1$, using an adaptation level of 1 causes the weight changes to be too large. The objective is to have the adapted Adalines provide their new response with some moderate confidence level. The parameters suggested above meet this requirement.

A deadzone value of zero is used. This means that only those Adalines actually accepted in trial adaptations will have their weights changed. There could be Adalines of very low confidence that are not accepted during the trial reversals. These Adalines are not adapted to provide a minimum confidence. (Note: A nonzero deadzone could be used. The simulation results presented in this paper used a zero deadzone. To use a nonzero deadzone, the master controller sends a special command to the Adalines of a layer after the trials are completed. The low confidence Adalines then adjust their weights to provide their current binary response with confidence level equal to the deadzone. Do this by using $\eta = 1$ and L = δ in Equation 1.1.)

The minimal disturbance principle demands one final consideration. The confidence levels of some Adalines in a layer may be quite high for a given input pattern. If such an Adaline is adapted, it will require a large change in its weight vector. The minimal disturbance principle suggests it is better to let the next layer assume responsibility for correcting output errors than to make such a large weight change. Experience with the algorithm confirms the merit of this idea.

There are two ways to limit the number of Adalines considered for trial adaptation in a layer. One way is to set a maximum value for the adjust signal. This prevents Adalines above a given threshold confidence from participating in trial adaptations. A second implementation is to allow only a set fraction of the total number of Adalines in a layer to be trial adapted. If a layer has ten Adalines and only the five least confident ones

are allowed to participate in trials, the set fraction is one-half.

If a nonzero deadzone is used, the second approach is preferable. It is difficult to know ahead of time how large the weights and, correspondingly, the confidence levels will to be to achieve a minimum confidence over a training set. This complicates the choice of a maximum value for the adjust signal. Experience indicates the fixed fraction idea provides an algorithm that works better over a larger set of cases. The value used in the simulations presented in this paper was one-half of the Adalines on a layer plus one if there were an even number of Adalines, and the big half if there were an odd number.

2.3 Usage

The implementation of MRII detailed in the previous section was simulated on a computer. The training of two-layer Madalines often failed to converge to a solution even on training sets for which there were known solutions. It was found that the failures were typified by one particular first-layer Adaline always being accepted during trial adaptations. This Adaline is called the "hinge" Adaline since the response of the Madaline as a whole was dependent on how this Adaline is adapted.

The training set divides itself into three subsets. The patterns in one subset are responded to correctly independent of the response of the hinge Adaline. Patterns in the second subset are responded to correctly when the hinge Adaline responds $+1$ while the third subset needs the hinge Adaline to respond -1 to provide a correct response. Usually, only one output bit is incorrect when an error occurs and the second and third subsets are small. The network is very near a global solution. Whenever a pattern from the second or third subset is presented and an output error occurs, the hinge Adaline has the lowest confidence in the first layer. The hinge Adaline also corrects the Madaline's response when it is trial adapted. Thus, the hinge Adaline is always trial adapted first and is always accepted for adaptation whenever an error occurs. Unfortunately, the second and third subsets are not linearly separable from each other. The dynamics of the algorithm are then exactly those of a single Adaline trying to separate a not linearly separable set.

This type of behavior was also noted to occur with the Ridgway Madaline. Glanz [3] reports that Hoff suggested using "activity levels" to force Adalines not being adapted to be adapted. Prior to discovering these notes by Glanz, the author came up with the concept of "usage." Usage is a way to modify the apparent confidence level of an Adaline.

A way to break the cycle of always adapting the hinge Adaline is to make its confidence

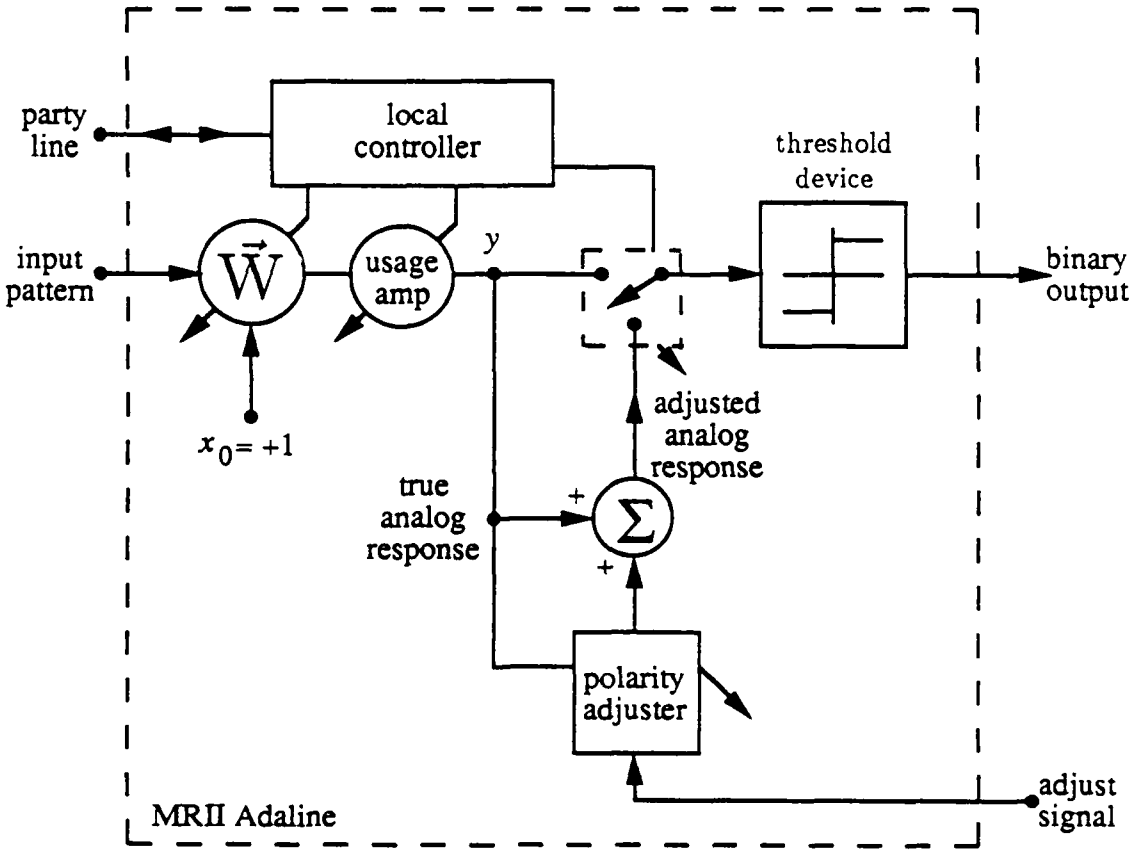


Figure 2.4: An Adaline modified to implement the usage concept.

not be the lowest when trial adaptations occur. This forces another Adaline to be trial adapted before the hinge Adaline. In this way, the rest of the Adalines on the layer can be forced to help the hinge Adaline separate the two problematic subsets. Usage modifies the minimal disturbance principle by requiring a spreading out of responsibility among all the Adalines.

To implement the usage concept, pass the true analog response through a variable gain amplifier. Set the gain to a value proportional to the number of times that particular Adaline has been accepted for adaptation. This gain can be set by the Adaline's local controller if it counts how many times it receives an adapt command from the master controller. A usage modified Adaline is shown in Figure 2.4.

Deciding exactly how to set the gain of the usage amplifier remains the most empirical part of the MRII algorithm. A formula which works well in simulations is:

$$\text{gain} = 1 + \frac{\text{adaptation count}}{\text{MULT} * N}$$

where N is the number of patterns in the training set and "MULT" is a multiplier value. Experience indicates MULT = 5 is a good choice. The idea here is to not let the usage gain get too large before the network really gets into a cycle.

This chapter presents a high-level, block diagram implementation of the MRII algorithm. The implementation has several advantages. Communications within the network are held to a minimum. No analog value, such as a weight or confidence level, needs to be communicated outside of the local Adaline environment. A non-output Adaline does not need to know its effect on the Madaline's output response; this information is passed indirectly by the master controller. For this reason, a party line only needs to span a single layer instead of being connected to all units. During trial adaptations, the master controller does not even need to know which Adaline(s) are participating in a given trial since the confidence level sorting is performed automatically by the adjust signal. Trial adaptations are performed without changing the weights. The weights are not changed until the trial is accepted, thus, no memory for storing the old weights is required. The implementation insures the number of trial adaptations performed on a given layer grows as the number of Adalines on the layer instead of exponentially. The cost of these advantages are minor. A relatively complicated local controller is required for each Adaline. This controller is the same for all the Adalines and is easily replicated using VLSI techniques. Its logic is fixed and does not need to be changed during the course of training or even when the network size is changed. The master controller should be programmable to handle different network sizes. The master controller determines the maximum number of Adalines trial adapted at one time and how many of the Adalines on a given layer are allowed to participate in the trials. This information can be programmed into the master controller and could even be allowed to change during the course of training. This results in an algorithm more complicated than presented here, but it is easy to do with implementation presented. Within the implementational framework presented then, the algorithm has "room to grow." The next chapter shows some results of simulations using MRII as defined by the details and heuristics presented in this chapter.

Chapter 3

Simulation Results

This chapter presents the results of using MRII to train various Madaline networks. Madalines are used to solve several problems including an associative memory problem and an edge detection problem. An investigation of how well MRII trains networks to learn arbitrary input/output mappings is also presented. This latter investigation also includes a study of the generalizing properties of MRII. Results of these experiments follow a presentation of the performance measures used to evaluate the algorithm.

3.1 Performance Measures

To evaluate a learning scheme, a set of criteria by which to measure performance needs to be established. Of primary concern, is whether the algorithm converges to a solution, and if it does, how fast. If it doesn't converge, some measure of how close to a solution the algorithm comes is needed. In either case, it is desirable to know how well the resulting trained network responds to inputs that are not in the training set. This is the issue of generalization. This section defines the measures used to evaluate MRII's performance.

Before actually addressing the measures used, the experimental technique used for the simulations is presented. Training begins with some set of initial weights in the Madaline network. The simulations randomize these initial weight values. Each individual weight is selected independently from a distribution that is uniform on the interval $(-1, +1)$. The training patterns are presented in random order. At each new presentation, each pattern is equally likely to be the one presented. The selection process can be visualized as pulling one pattern out of a grab bag containing all the patterns and returning the pattern to the bag

after each presentation. At regular intervals during training, all of the patterns are presented to the network one after another to determine how many of them are being responded to correctly. No adaptations are performed during these intermediate performance checks, which are called "checkpoints." If the responses to all patterns in the training set are correct, then convergence is said to occur. Finally, the results are presented as the average of an ensemble of training attempts, each attempt having a new set of randomized initial weights.

The first performance measure to be considered is one that counts how many steps it takes the algorithm to reach a solution. For a Madaline implemented in hardware, the process of presenting a pattern and getting a response will take almost no time. If the response is correct, the network immediately goes on to the next pattern. If the response is incorrect though, the network undergoes a series of trial adaptations until the output is corrected. The trial adaptation procedure takes up the majority of the training time. As convergence is reached, most pattern presentations require no adaptations. Thus, the reasonable measure of training time is the number of pattern presentations that require adaptations be performed. This is a good measure of the time an actual hardware implementation would require for training. Furthermore, the network only learns from its mistakes. A pattern presentation that requires adaptations be made is called a "learning opportunity." By measuring network performance against the number of learning opportunities, a measure of the algorithm's efficiency as it proceeds towards convergence is obtained.

The most interesting Madalines are those with more than one output bit. Ultimately, convergence requires that all bits of all output patterns be correct. In measuring performance prior to convergence, one has two choices. One can count the number of input patterns for which the output response is completely correct or one can count the total number of output bits that are correct over the entire training set. One would expect that the number of input patterns that have a completely correct response will not increase until almost all of the output bits are correct. A distinction is made then between pattern rates and bit rates. The "pattern training rate" is the percentage of patterns in the training set for which the Madaline makes a completely correct output response. For example, if the training set has 50 patterns in it and at some point during training the Madaline's output is completely correct for 24 of these patterns, the pattern training rate is 48%. The "bit training rate" is the percentage of the total number of output bits that are correct when a check of the Madaline's performance on the training set is made. For example, if a Madaline has three output bits, the training set has 50 patterns, and 123 output bits are correct at

a checkpoint, the bit training rate is said to be 82%.

It is a fact that MRII does not always arrive at a solution. Sometimes it takes a very long time for a solution to occur from a particular set of initial weights. At other times, the algorithm gets trapped in a limit cycle and does not proceed to a solution. In performing the simulations in the following sections, the problem to be solved is attempted many times starting from new initial weights each time. It is necessary to set a limit (based on experience) to the number of learning opportunities that are allowed before the particular attempt is abandoned. If convergence has not occurred by the time this limit is reached, it does not necessarily mean convergence would not occur if training were continued. This must be remembered when a convergence rate is reported. The convergence rate is the percentage of training attempts that reached a solution within some allowed limit of learning opportunities. When a training attempt converges, the number of learning opportunities used to come to a solution will be less than the allowed limit. For the training attempts that converge, an "average learning opportunities to converge" is reported. Suppose three training attempts result in convergence twice within an allowable limit of 1000 learning opportunities. If one instance of convergence used 666 learning opportunities and the other used 222 learning opportunities, the convergence rate for the ensemble of three attempts is reported as 67% and the average learning opportunities to converge is 444.

Learning curves are a way to present the average performance of a Madaline as it is trained. Learning curves use information that is gathered at the checkpoints, specifically, the pattern training rate and the bit training rate at those points. Suppose an ensemble of 50 training attempts is made. During each training attempt, the checkpoints are set after every 100 learning opportunities. (A checkpoint is also performed prior to the start of training.) If 36 of the training attempts result in convergence, checkpoint data for these 36 cases will be averaged to compute the learning curve. A "pattern learning curve" plots the average pattern training rate versus the number of learning opportunities. Data points for the plot occur at the checkpoints (every 100 learning opportunities) and the plot will connect the data points with a straight line segment. Similarly, a "bit learning curve" plots the average bit training rate versus learning opportunities.

The training attempts that converge are used to generate learning curves that typify the algorithm's performance on the particular training problem. For the cases of nonconvergence, a measure of how close the algorithm came to convergence is desired. With extra circuitry, it is possible to build a Madaline that can save its weights at checkpoints where training performance is especially good. The learning exhibited by MRII is not a monotonic

process. This is due to the fact that weight changes are made based only upon the particular pattern being presented rather than on a global assessment of the effect of weight changes on the entire training set. Some adaptations cause regression of global performance. The ability to save weights at intermediate stages minimizes this effect. The important criterion then is the best performance exhibited at any stage of training. In a case of nonconvergence, the highest bit training rate observed at any checkpoint is identified. This best rate is averaged over all cases of nonconvergence in the ensemble of training attempts to report an "average best nonconverged bit training rate."

3.2 An Associative Memory

An associative memory associates an input pattern with an output pattern. A need for such an application arises in the adaptive descrambler portion of a translation invariant pattern recognizer reported in Widrow, et al [11]. The basic structure of the system is shown in Figure 3.1. A retinal image containing a possibly translated figure is presented to an invariance box. The invariance box outputs a pattern that is unique to the figure presented and does not change when the figure is translated on the retina. The descrambler associates this translation invariant representation of the figure with the original figure in a standard position. The translation invariant input pattern acts as a key that unlocks the required desired response. Most associative memories have the ability to perform well with incomplete keys or keys with errors in them. This is not required of this application. The requirement here is for the descrambler to act as a simple lookup table.

The structure used in this application is a 25 input, 25-feed-25 Madaline. There are 36 patterns in the training set. Three different training sets are used, each representing a different invariance box. The descrambler is made adaptive because the exact form of the outputs from the invariance box is dependent on the particular problem. The use of different training sets shows the general ability to associate patterns with different invariant representations.

The learning curves for patterns and bits are presented in Figure 3.2. The pattern learning curve doesn't show an appreciable increase in performance until almost all the bits are learned. This is because a pattern is not considered correct until all its bits are correct.

The algorithm always reaches convergence when training on this application. It also reaches a solution in a reasonable amount of training, only needing to adapt on each pattern about 15 times to insure convergence. The network is not being tasked heavily by this

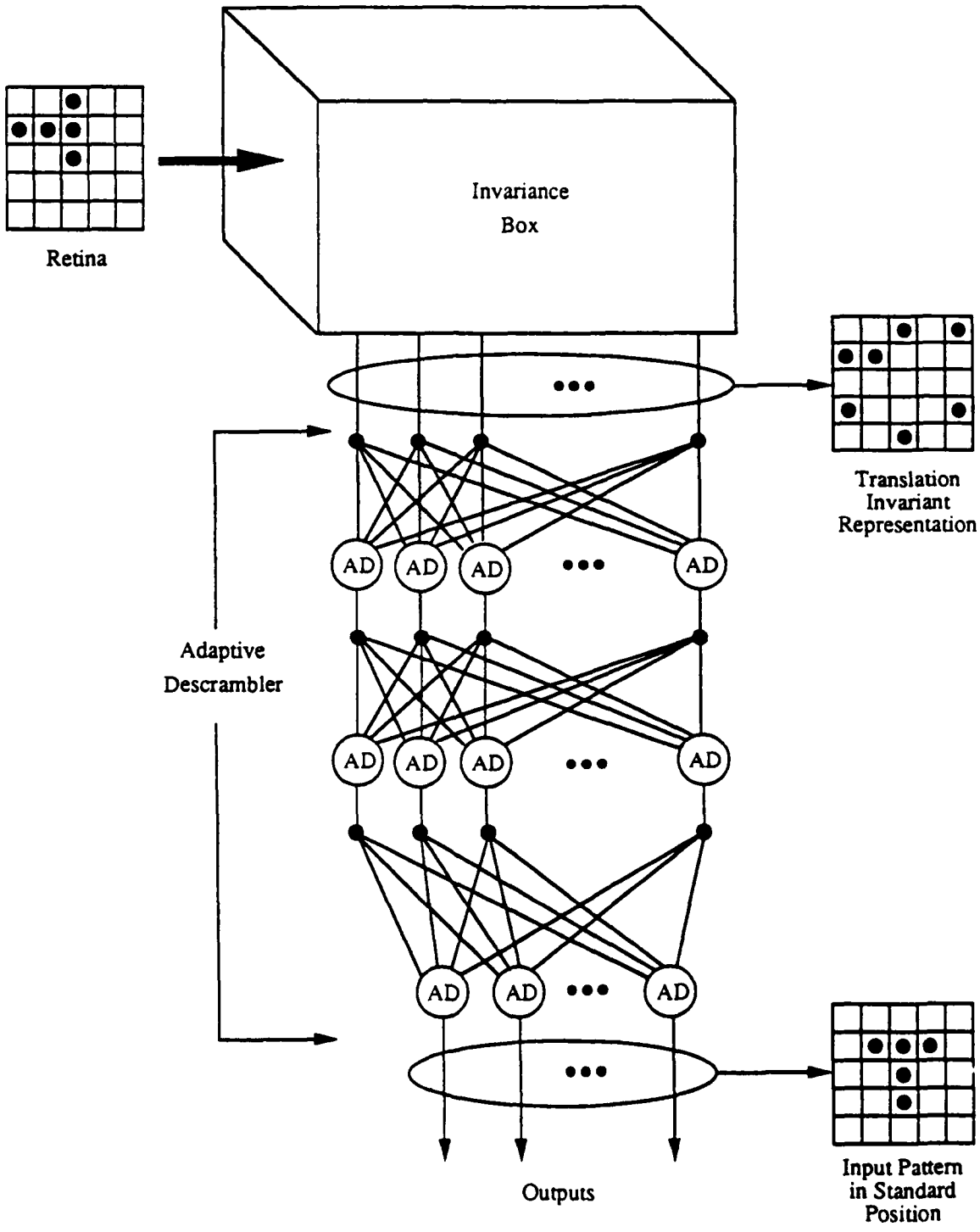


Figure 3.1: Diagram of a translation invariant pattern recognition system. The adaptive descrambler associates a translation invariant representation of a figure to its original representation in standard position.

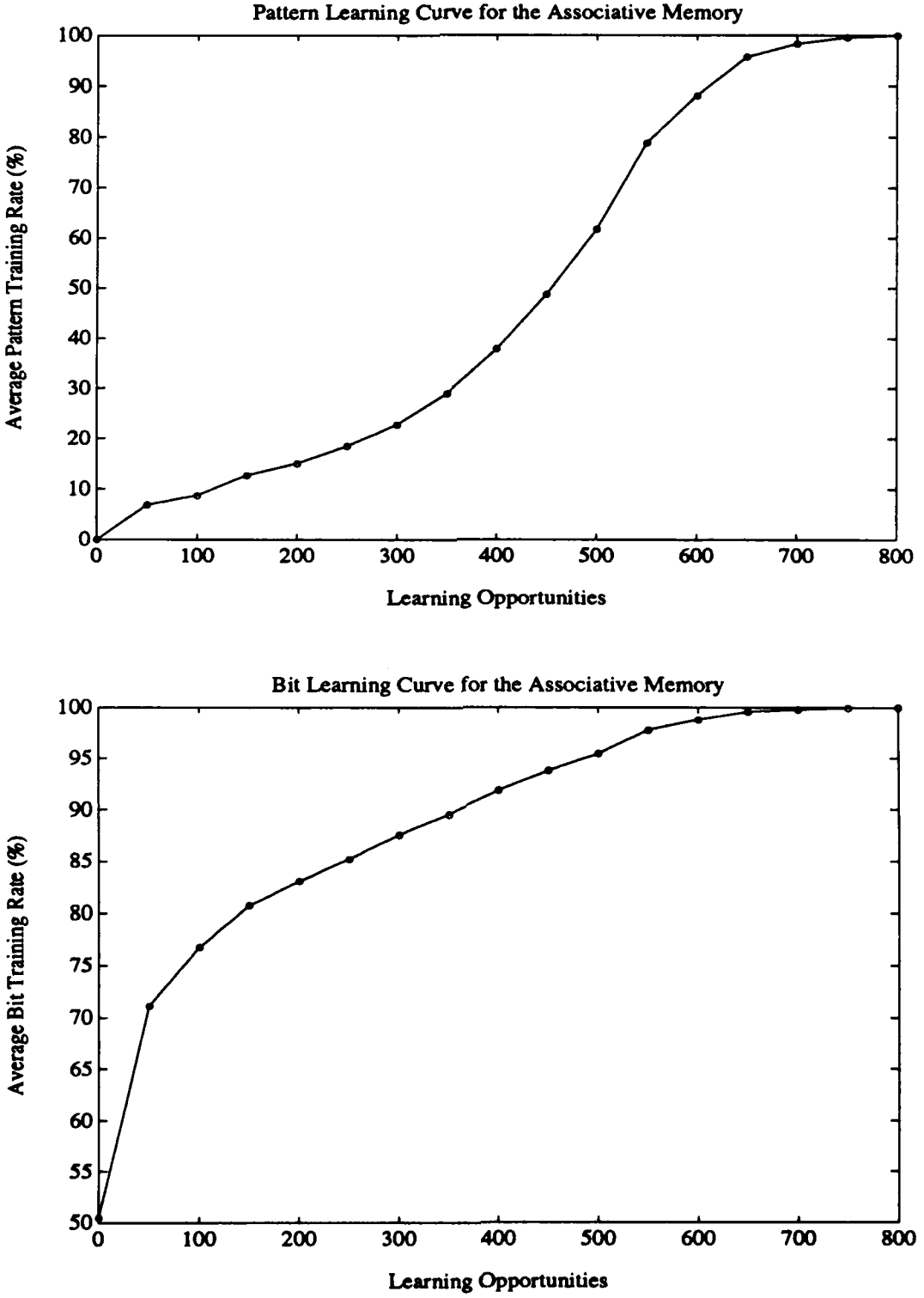


Figure 3.2: Learning curves for patterns and bits for the adaptive descrambler associative memory application. Average of training 100 nets. Data points are marked by "o".

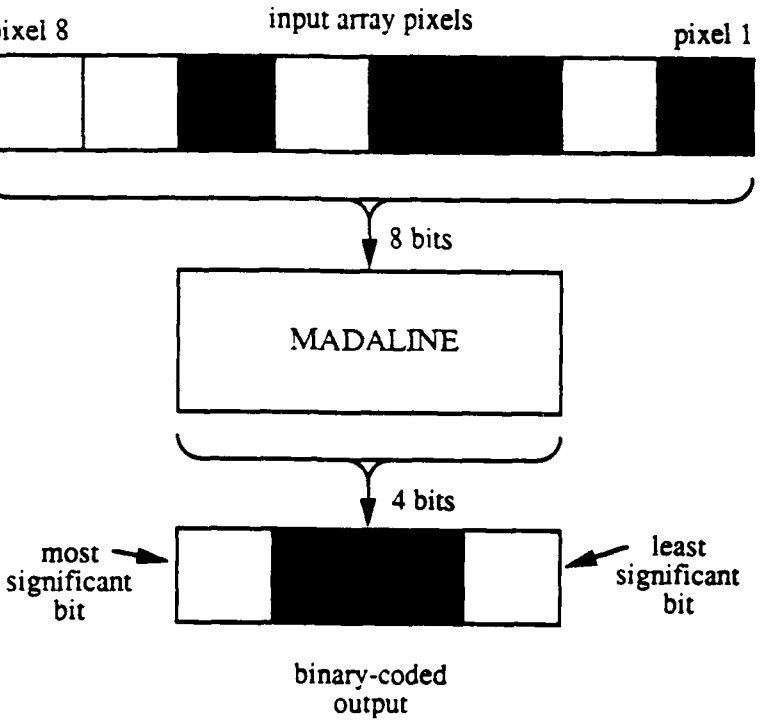


Figure 3.3: A component of an edge detector system. The Madaline outputs the position of the first black pixel in a binary code.

example. While the capacity of layered feed-forward Madalines is not known, one can suspect it exceeds 36 for the structure being trained here. The example does show that MRII is capable of training networks with inputs of relatively large dimension and with many units on both the first and output layers.

3.3 Edge Detector

Image analysis techniques often require extraction of certain features from a scene, such as edges. A Madaline that can be used as a component of an edge detection system was simulated and trained using MRII. The problem is illustrated in Figure 3.3. The Madaline receives its input from a linear array of pixel elements. The array has a horizontal orientation. The task is to detect the first "on" pixel scanning from left to right and report its position using a binary code. The array has eight elements and the case of no pixels being on must be allowed for. This means there are nine possible outputs for the Madaline, requiring four output Adalines to represent them. The minimum possible network then to

Edge Detector Performance			
network architecture	convergence rate (%)	average learning opportunities to converge	average best nonconverged bit training rate (%)
4-feed-4	52	545	99.524
5-feed-4	61	422	99.775
6-feed-4	89	282	99.849

Table 3.1: Performance for the edge detector problem. Averages are for training on 100 cases beginning with random initial weights.

solve the problem is an 8 input, 4-feed-4 Madaline.

The training set consists of all 256 possible sequences of "on" and "off" pixels represented as sequences of $+1$s and -1s. Besides the minimal network, 8 input, 5-feed-4 and 6-feed-4 Madalines are also trained to solve the problem.

Table 3.1 lists the simulation results for the different network architectures. One hundred cases starting from different initial conditions are trained for each network architecture. Training is terminated after 2000 learning opportunities if convergence is not reached before this limit. The convergence rate and average learning opportunities to convergence are listed for each architecture trained. For those cases where convergence was not reached, the average best nonconverged bit training rate is listed.

When the algorithm fails to converge, usually all but a couple of input patterns are learned correctly. These unlearned patterns usually have only one output bit per pattern in error. This is why the average best nonconverged bit training rate is so high. The network arrives at a solution that satisfies all but a few output bits out of the 1024 total. This failure to converge led to an investigation of the algorithm dynamics to find out why convergence doesn't occur. The next chapter presents an example of a failure mode identified during that investigation. All failures of the algorithm to converge on the edge detector problem are of this type.

Further examination of the simulation results reveals an expected result. There is a noticeable trend as the trained net becomes larger in size than absolutely necessary to solve the problem. The convergence rate increases and training time, as measured by the number of learning opportunities to converge, decreases.

The use of a network larger than the minimum required is referred to as "overarchitecturing." Overarchitecturing in the case of networks with two layers of adaptive units

means using more first-layer units than necessary. The number of output units is fixed by the problem definition, i.e., the number of output bits required for pattern classification. As the network is overarchitectured, there are more choices of a set of hidden patterns that is separable in the required ways by each of the output units. This makes a given problem easier to solve. Thus, it is expected that the convergence rate would increase and the training required to reach convergence to decrease.

This sample problem does not address the issue of generalization. Generalization is the ability of a network to respond correctly to patterns that are not in the training set. In this example, the entire set of possible patterns is presented to the network during training. The issue of generalization and the effect of overarchitecturing on generalization performance is addressed in the next section.

3.4 The Emulator Problem

One of the intended applications of neural networks is to learn the nonobvious relationships between the inputs to a decision process and the resulting decision. It is a relatively simple problem to write down a boolean logic function for the edge detector of the previous section. Each output bit is a relatively simple function of the eight input bits. The ease with which this can be done makes the problem a simple one to program on a computer. In similar fashion, one can write logical relationships for each output bit of the associative memory problem addressed earlier. Due to the dimensionality of that problem, it would not be easy. It is difficult to program this problem as a relationship between input bits and output bits. Thus, the function the Madaline performs in the associative memory case is nonobvious. What ability does MRII have to train networks to learn nonobvious, perhaps arbitrary, input/output relationships?

A structure to investigate this question is shown in Figure 3.4. The idea is to train an adaptive net to emulate a fixed net. The two nets receive the same inputs and have the same number of output units. During training, the fixed net acts as a teacher by providing desired responses to the adaptive net. The fixed has its weights set randomly to provide a wide range of arbitrary input/output mappings for the adaptive net to learn. The adaptive net has its weights initialized to some other random set of values. MRII is used to train the adaptive net to respond just like the fixed net to a given input.

If the adaptive net has the same architecture as the fixed net, then at least one set of weights exist that allows the adaptive net to emulate the fixed net perfectly, the set in

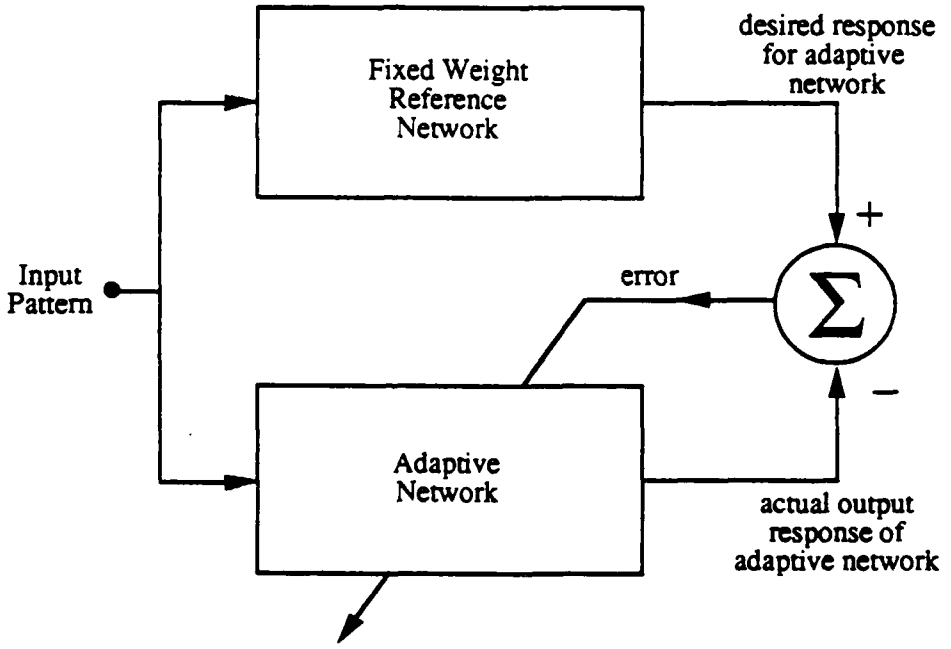


Figure 3.4: Training a network to emulate another. The fixed net provides desired responses for the adaptive net.

the fixed net. With this structure, one is assured that the problem being presented to the adaptive net is solvable.

The issue of generalization is easily addressed by this structure. Training is conducted on a subset of the possible input patterns. After training is complete, patterns not in the training set are presented in parallel to the fixed and trained nets and their responses compared.

The issue of overarchitecturing can also be addressed. There is no need for the adaptive net to have the same number of first-layer units as the fixed net. It could have more. By zeroing the weights of the unneeded units, the perfect solution offered by the fixed net can still be obtained. The effect extra units have on generalization performance, convergence rate, speed of convergence, and sensitivity to weight errors or drift are easily checked with the test configuration of Figure 3.4.

The example presented here is a 16 input, 3-feed-3 fixed net. Adaptive nets with 3, 6, and 9 first-layer Adalines are trained. Two different size training sets are used. The smaller set has 650 patterns and the larger set has 1500 patterns. The patterns are selected randomly from the set of possible input patterns which numbers $2^{16} = 65,536$. Training is allowed

Emulating a 16 input, 3-feed-3 Fixed Madaline					
adaptive net architecture	convergence rate (%)	ave learning opportunities to converge	generalization bit rate(%)	generalization pattern rate(%)	ave best nonconverged bit trng rate (%)
650 Pattern Training Set					
3-feed-3	58	22940	98.11	95.98	89.28
6-feed-3	72	20860	96.70	92.20	88.84
9-feed-3	80	16120	95.43	88.66	90.68
1500 Pattern Training Set					
3-feed-3	39	50860	99.20	98.32	92.43
6-feed-3	65	46610	98.68	96.92	90.33
9-feed-3	65	37780	98.13	95.36	89.31

Table 3.2: Training and generalization performance for the emulator application. The fixed net was 16 input, 3-feed-3.

to continue until convergence or until 100,000 learning opportunities occur. After training is stopped, generalization is checked. This is reported with a "generalization pattern rate" and "generalization bit rate" similar to the corresponding training rates. Instead of doing a "checkpoint" on the training set, the generalization rates are obtained by doing a checkpoint on the balance of the patterns in the input space that are not in the training set. The results are shown in Table 3.2.

As in the case of the edge detector, overarchitecturing improves training performance. The number of training attempts that reach convergence increases as extra units are added to the adaptive net. The number of learning opportunities to reach convergence decreases as the number of first-layer Adalines increases. This trend was noted before with the edge detector and is not surprising. The generalization results though are surprising.

Generalization decreases with overarchitecturing. This is expected. With a greater number of possible ways to select a hidden pattern set that can be used to solve the training set, it seems more likely that a set will be selected that solves only the training set and not the entire space as mapped by the reference network. The amount of decrease is very slight though and this is surprising. Consider an adaptive net with nine first layer units versus one with only three. The net with nine units has a hidden pattern space with 512 patterns available in it. The minimal architecture can solve the problem with only eight. The fact that the larger net arrives at a solution that preserves generalization performance, to the degree exhibited in the results, is counterintuitive. Although no firm conclusions

can be drawn from these experimental results without theoretical back-up, the results are representative of a considerable amount of simulation experience.

As expected, the smaller training set is learned quicker and with a greater convergence rate than the larger set. There is also the expected decrease in generalization performance. This decrease is slight, becoming significant only with the generalization pattern rate for the 9-feed-3 architecture. This decrease is greater for the generalization pattern rates than for the generalization bit rates. The results indicate that 650 patterns are sufficient to define the type of problem the fixed net can present. The 650 patterns are sufficient to insure good generalization, at least as far as bits are concerned, even from the highly overarchitectured 9-feed-3 adaptive net. This 650 pattern training set represents about one percent of the available patterns in the space. It is possible that an even smaller training set might maintain good generalization performance with even faster and more sure training. In general, the minimum number of patterns required in the training set to insure good generalization performance has not been determined.

Use of the smaller training set and larger adaptive nets can save a lot of training time. The 9-feed-3 net with the smaller training set converges twice as often with one third the number of learning opportunities as the 3-feed-3 net trained on 1500 patterns. The difference in generalization performance is a modest decrease.

Not all training attempts reach convergence. Table 3.2 reports the average best non-converged bit training rate for those cases that do not converge. Training set size and architecture of the adaptive net have little effect on this measure of performance. While not reported in the table, generalization testing was also performed after training for the nonconverged cases. Generalization performance is very close to the performance on the training set at the cessation of training. To check how well generalization tracks training performance, a generalization check was done at each of the training checkpoints during a particular training attempt. A typical result is plotted in Figure 3.5. The adaptive network is 9-feed-3, the one with the worst expected generalization performance, being trained on the 1500 pattern training set. The plot shows that generalization and training performance track each other closely at all levels of performance. This allows one to expect that generalization performance will be good anytime the training performance is good, if a sufficiently large training set is used. The plot shows that little added generalization performance is obtained during the 9000 learning opportunities that led to final convergence.

The plot in Figure 3.5 is a learning curve for an ensemble of size one. It is typical of MRII's performance during any particular training attempt. MRII's performance rises

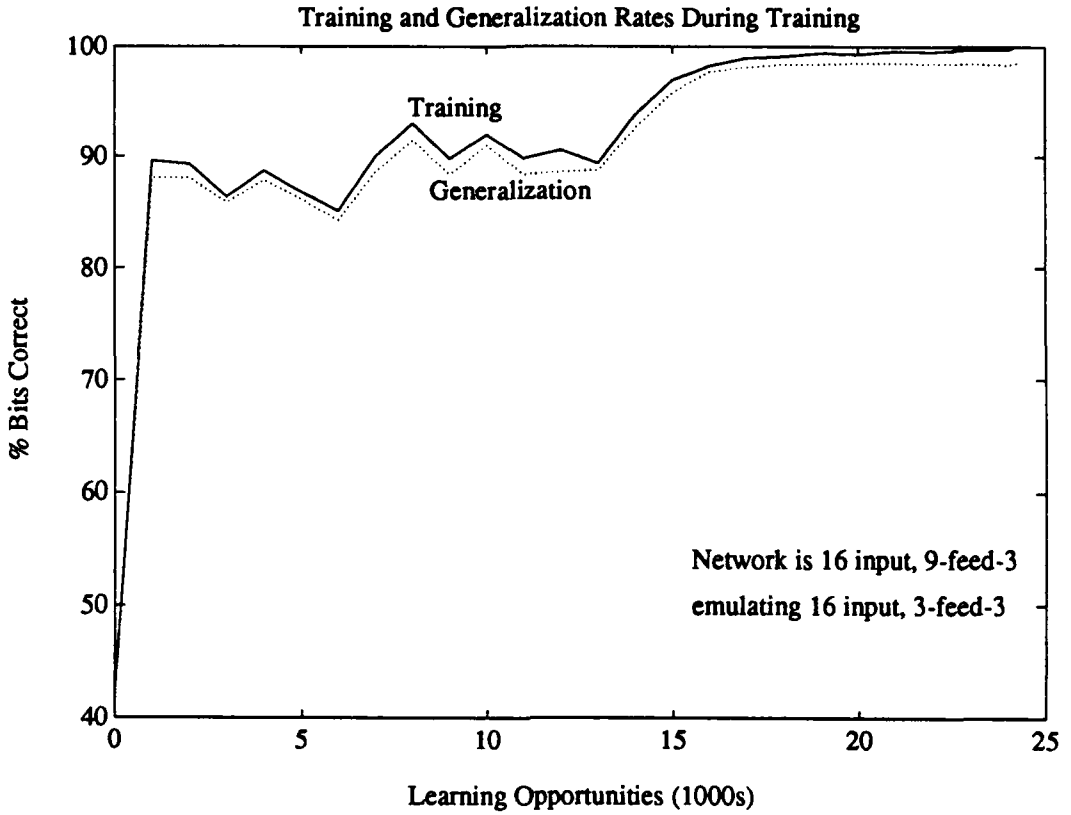


Figure 3.5: Tracking of generalization and training performances. Network is a 16 input 9-feed-3 emulating a 3-feed-3 fixed net.

quickly and then levels off. Further improvements take a relatively long time to occur and do not happen monotonically.

For those cases of nonconvergence, the final improvement towards convergence does not happen in the limit set for learning opportunities. Unlike the failures to converge identified for the edge detector and characterized in the next chapter, there is no indication here that convergence wouldn't occur if enough time were allowed. The failure mode identified in the next chapter may be responsible but is unrecognizable. Due to the larger dimensionality of this problem, specific reasons for nonconvergence cannot be identified.

Chapter 4

A Failure Mode of MRII

The previous chapter reports that the MRII algorithm does not always converge to a solution when starting from a given set of initial weights. Examination of some instances of nonconvergence reveals a failure mode of MRII. This chapter describes the failure mode by examining a particular instance of it. This simple example shows that a modification of the algorithm will be required to handle the failure mode. Some possible modifications are presented. The failure mode presented here may not be the only one that exists. A variant of the failure mode is described in the last section of the chapter. The author does not exclude the possibility of existence of other, yet unidentified, failure modes.

A secondary purpose of this chapter is to provide the reader some insight into the dynamics of the algorithm. The example used in this chapter is simple enough to represent in two dimensions, allowing presentation using graphical methods. The reader can see what happens to the mappings from input pattern to hidden pattern to output pattern as trial adaptations and weight changes are made. While the chapter highlights a case where the algorithm fails to proceed to a solution, its study leads to a better understanding of the methodology used by the algorithm.

4.1 The And/Xor Problem

The particular problem used for this example is called the and/xor problem. The challenge is to train a 2-input, 2-feed-2 network to give the logical "and" of its inputs as its first output, and the exclusive-or of its inputs as the second output. The desired input/output mapping is presented as a truth table in Table 4.1.

TRUTH TABLE			
input pattern		desired response	
x_1	x_2	d_1	d_2
+1	+1	+1	−1
+1	−1	−1	+1
−1	+1	−1	+1
−1	−1	−1	−1

Table 4.1: Truth table for the and/xor problem.

A solution for this problem is shown graphically in Figure 4.1. In the figure, the linear decision lines of the various Adalines are identified along with the weight vectors which define them. To allow easy reference to these weight vectors a double superscript is used on the weight vector symbol. For example, $\vec{W}^{21}$ refers to the weight vector of the first Adaline on the second (output) layer. As there is only one hidden layer in this network, the superscript on $\vec{H}$ and its components has been dropped.

This particular problem forces the Adalines of the first layer to perform double duty. They must provide a hidden pattern set separable by the second output Adaline to do the exclusive-or function. The hidden pattern set must also be separable by the first output Adaline to allow realization of the and function. Thus, there is an implicit requirement for the output Adalines to cooperate with each other on settling upon a mutually satisfactory hidden pattern set. The whole goal of MRII is to implement a method to facilitate this cooperation.

For this particular problem and for some of the problems presented in Chapter 3, there are states for the network from which MRII cannot proceed to a solution. The next section shows an example of a limit cycle that MRII cannot escape.

4.2 A Limit Cycle

Consider a network with the following set of weights in it:

$$\vec{W}^{11} = [.3, -.15, -.3]^T$$

$$\vec{W}^{12} = [-.15, .05, -.05]^T$$

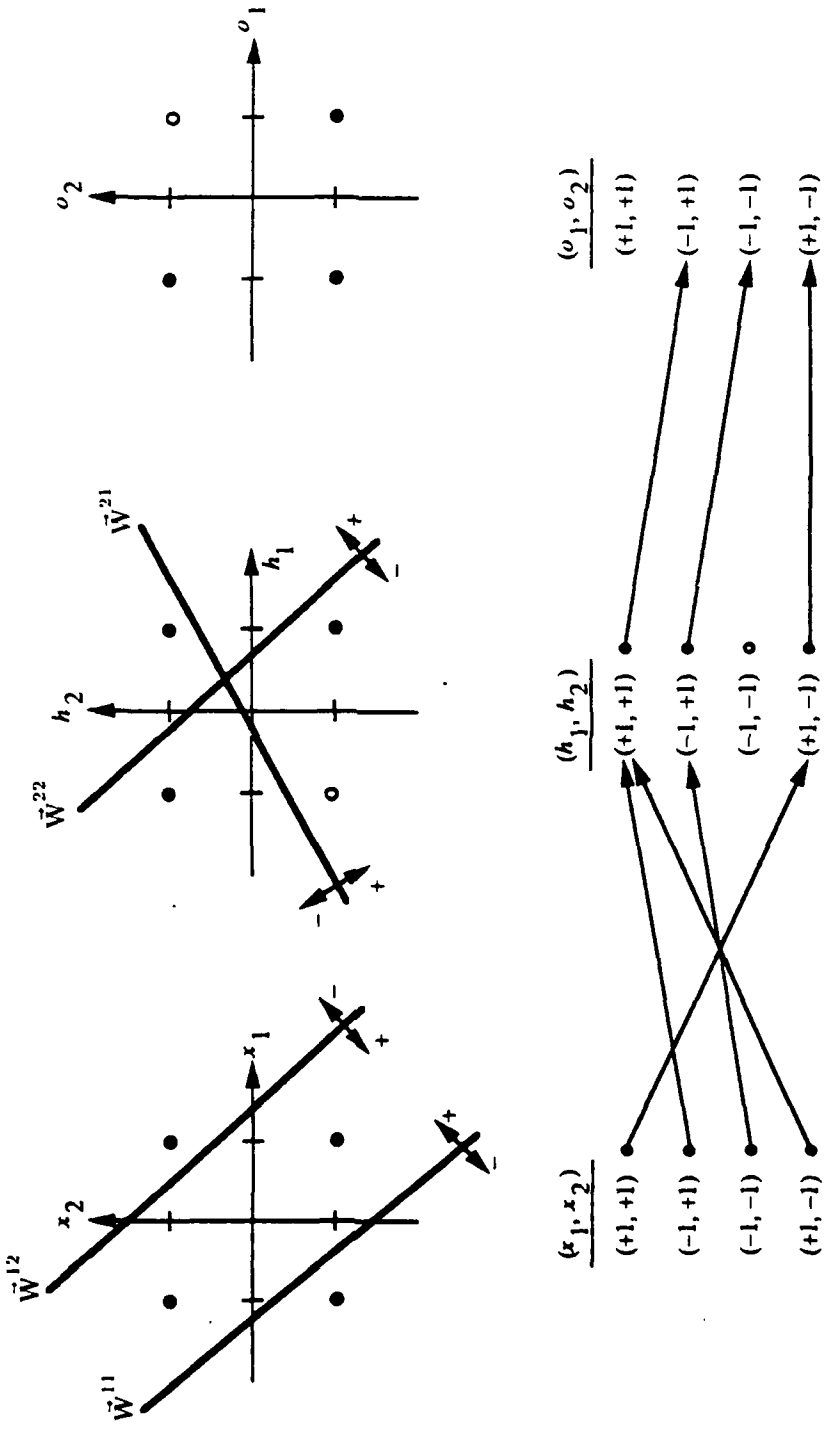


Figure 4.1: Graphical representation of a network that solves the and/xor problem.

State 1					
input pattern		hidden pattern		output pattern	
x_1	x_2	h_1	h_2	o_1	o_2
+1	+1	−1	−1	+1	−1
+1	−1	+1	−1	−1	−1 *
−1	+1	+1	−1	−1	−1 *
−1	−1	+1	−1	−1	−1

Table 4.2: The input pattern to hidden pattern to output pattern mapping for **State 1**. Incorrect output responses are marked by *.

$$\vec{W}^{21} = [.15, -.15, .15]^T$$

$$\vec{W}^{22} = [-.15, .15, .15]^T$$

These weights were taken from an actual simulation result and occurred after several learning opportunities. They are rounded to make easier the algebra and graphing. The decision lines and mappings for this network are shown graphically in Figure 4.2. The mapping information is also shown in tabular form in Table 4.2. In the table, incorrect output responses are indicated by an asterisk. This set of weights and mappings is called **State 1**.

The network in **State 1** makes an error for two of the four input patterns. The two patterns produce the same hidden pattern and output response and also have the same desired response. Therefore, it makes no difference which of these two patterns is considered first. Until one of these error producing patterns is presented, no adaptations will take place since the responses for the other two input patterns are correct. When one of the two input patterns that result in an error is presented to the network, MRII will attempt to correct the output by trial adapting the first layer Adalines.

At this point, it is important to note that o_1 is already correct. MRII accepts a first-layer trial adaptation then, only if it completely corrects the output pattern. It will not accept a "trade" of errors in the bits of the output pattern. The Hamming distance between the output and desired response must actually be reduced.

For an error producing input, the first layer responds with the hidden pattern $h_1 = +1$, $h_2 = -1$. A trial adaptation can be thought of as moving this hidden pattern to another position in the hidden space and checking the decisions made by the output Adalines on this new point. The concept is shown graphically in Figure 4.3. The same information is shown

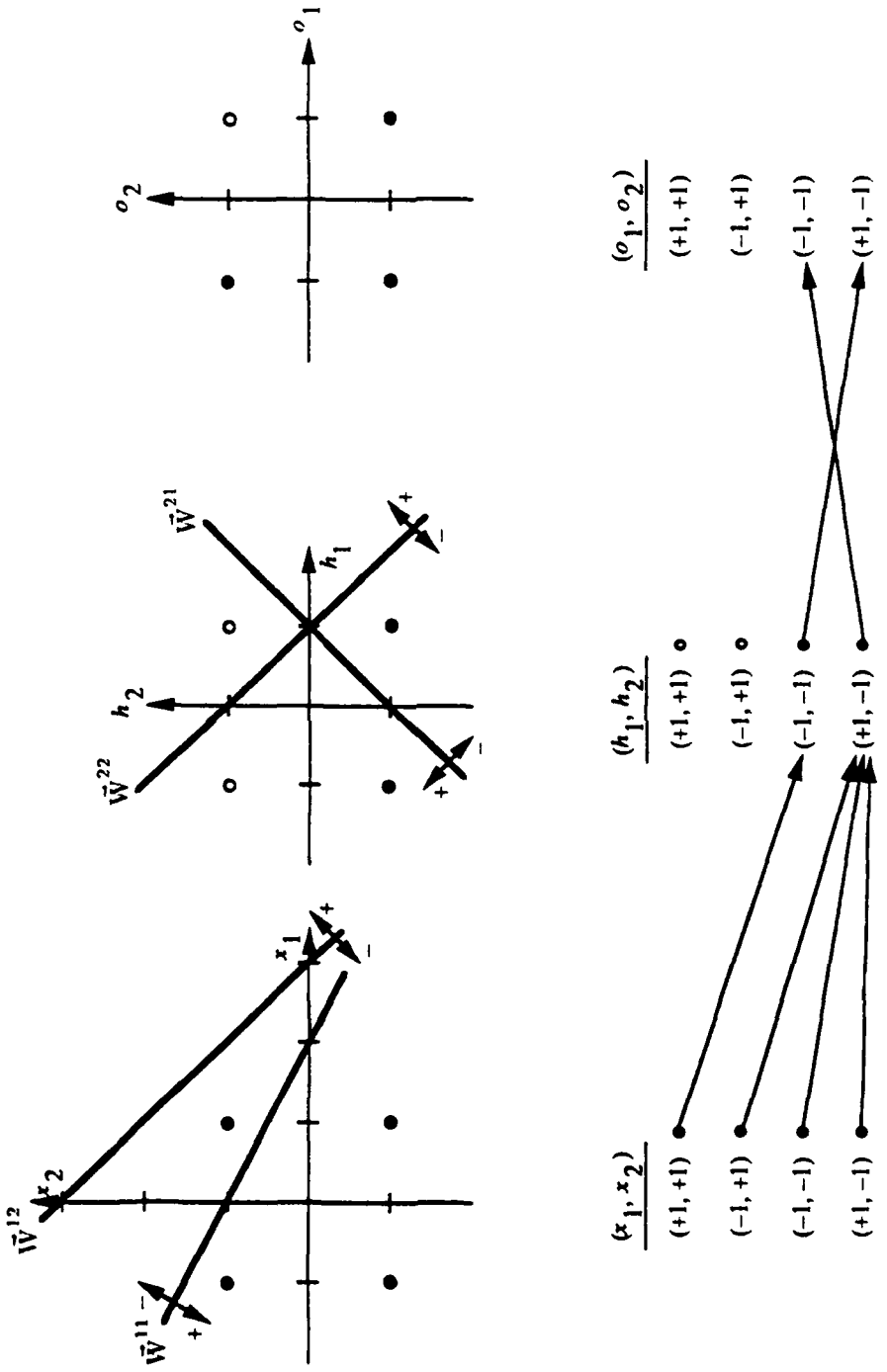


Figure 4.2: Graphical presentation of **State 1**.

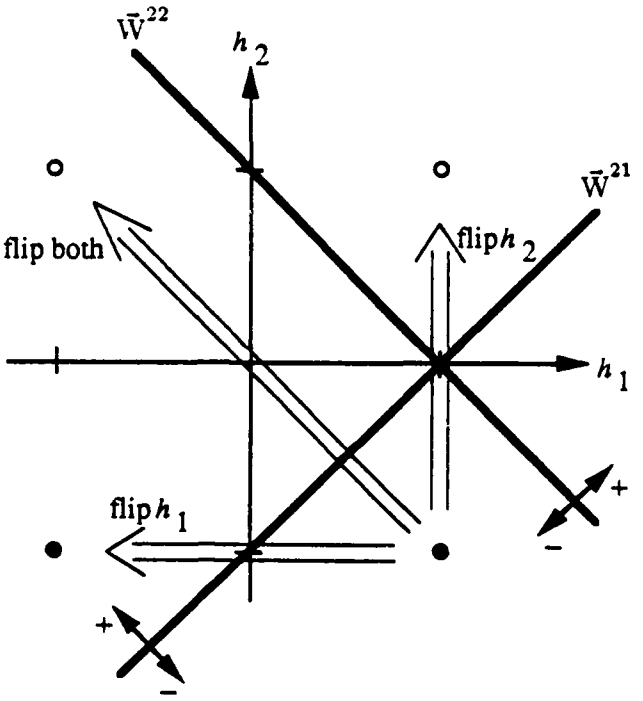


Figure 4.3: Graphical presentation of trial adaptations from **State 1**.

in tabular form in Table 4.3. The table shows that none of the possible trial adaptations completely correct the output, so they are all rejected. Failing to find an acceptable first-layer adaptation, MRII proceeds by adapting the second output Adaline to provide the desired response, $+1$, for its input, the hidden pattern $h_1 = +1$, $h_2 = -1$.

This brings the network to what is called **State 2**. The only change in the network has been to the second output Adaline. The hidden patterns used by the network have not changed. It is assumed the response to the hidden pattern $h_1 = -1$, $h_2 = -1$, has not been affected. The mapping information for the network in **State 2** is shown in Table 4.4. A repositioning of the second output Adaline's decision line in the hidden space to provide this mapping is shown in Figure 4.4.

As Table 4.4 shows, the network now makes an error for only one input pattern, $x_1 = -1$ and $x_2 = -1$. No adaptations are made by the network until this pattern is presented. The possible trial adaptations that can be made in this case have the same bad result that occurs from **State 1**. All trials result in o_1 becoming incorrect. This is because the hidden pattern is still the same as that considered in **State 1** and the decision line of the first

Trial Adapting for State 1				
Trial Adapt	h_1	h_2	o_1	o_2
none	+1	−1	−1	−1 *
flip h_1	−1	−1	+1 *	−1 *
flip h_2	+1	+1	+1 *	+1
flip both	−1	+1	+1 *	−1 *

Table 4.3: The results of trial adaptations from **State 1**.

State 2					
input pattern		hidden pattern		output pattern	
x_1	x_2	h_1	h_2	o_1	o_2
+1	+1	−1	−1	+1	−1
+1	−1	+1	−1	−1	+1
−1	+1	+1	−1	−1	+1
−1	−1	+1	−1	−1	+1 *

Table 4.4: Mapping information for the network in **State 2**.

output Adaline has not changed in getting to **State 2**. Therefore, all the trial adaptations are rejected. The second output Adaline again requires adaptation. Its input is the same as when it adapted out of **State 1**, but the error is opposite to what it was. Thus, the adaptation returns the network to **State 1**, basically undoing the previous adaptation. The network is in a limit cycle.

The astute reader will now question whether **State 2** is the only possible result of adapting out of **State 1**. Suppose the second output Adaline's response to the hidden pattern $h_1 = -1$, $h_2 = -1$ also changes with the adaptation out of **State 1**. Indeed, a plausible positioning for the second output Adaline's decision line is shown in Figure 4.5. This results in a **State 3** whose mapping is detailed in Table 4.5.

An examination of Table 4.5 shows there are now two input patterns that result in errors. MRII presents patterns in a random order. The pattern $x_1 = +1$, $x_2 = +1$ could be presented prior to the other error producing input. This input pattern maps to a different hidden pattern than considered previously. For this hidden pattern, the trial adaptation involving a reversal of h_2 corrects the output completely and is accepted. This trial adaptation is indicated on Figure 4.5. Thus, there exists a mechanism for escape

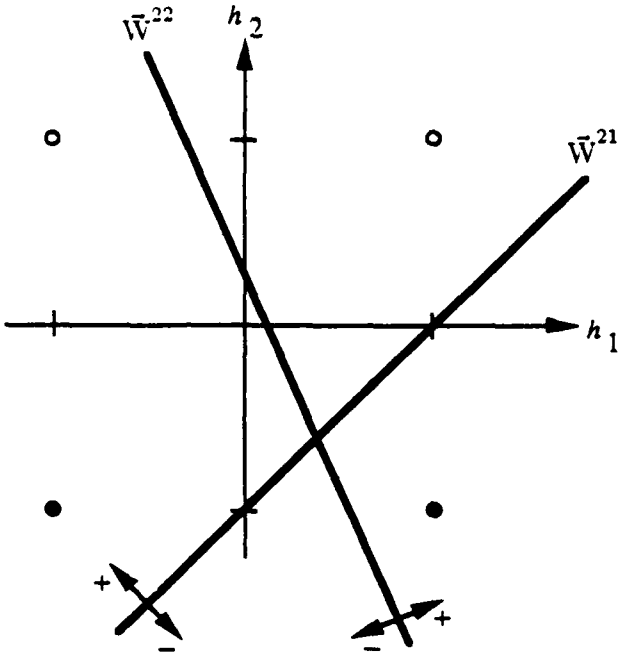


Figure 4.4: Separation of the hidden pattern space for **State 2**.

from the claimed limit cycle *if* **State 3** can be reached. Unfortunately, **State 3** cannot be reached as a few lines of algebra will now show.

MRII uses the modified relaxation method for performing its weight updates. Given that it is desired to adapt only when there is a binary error present, all of the adaptation procedures proposed by Mays [5] have the same basic form. Using k as an adaptation counter, all the weight-change formulas have the form,

$$\Delta\vec{W}(k) = \delta(k)\, d(k)\, \vec{X}(k)\,, \quad \delta(k) > 0.$$

Here, δ scales the adaptation step size and it is emphasized that it is not necessarily constant from adaptation to adaptation.

MRII adapts only when there is a binary decision error present. This is in accordance with the minimal disturbance principle. When following this procedure, the sign of the desired response and the sign of the analog error defined as, $\epsilon(k) = d(k) - y(k)$, are the same. Thus, even the famous Widrow-Hoff least means squares (LMS) algorithm has the form above. The LMS algorithm is usually written, $\Delta\vec{W}(k) = 2\mu\epsilon(k)\,\vec{X}(k)$. Here, the

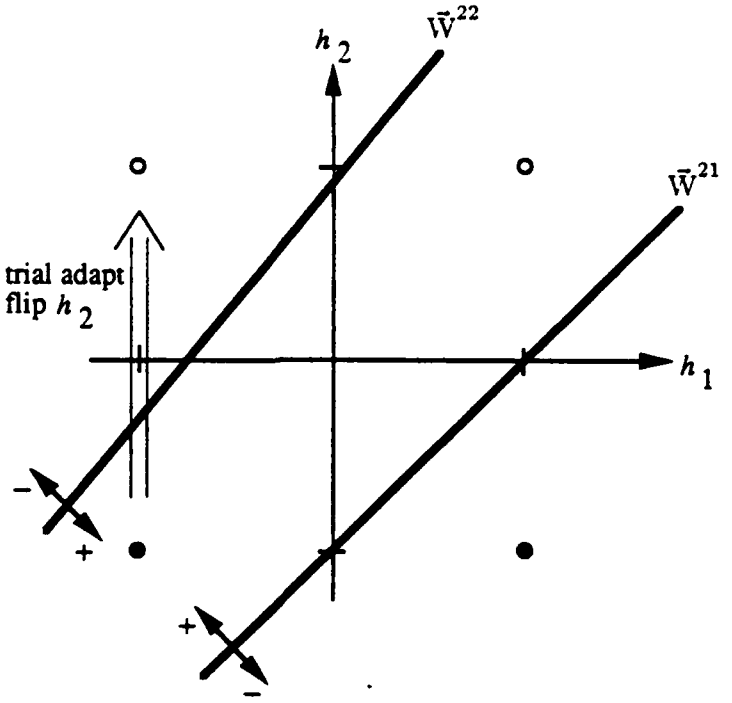


Figure 4.5: An output separation that would give **State 3**.

magnitude of $2\mu\epsilon(k)$ can be identified as $\delta(k)$. All the various weight update methods used to adapt Adalines differ only in how they choose $\delta(k)$. It will be shown that not only can **State 3** not be reached, but this unreachability is not dependent on the weight update rule used.

Under consideration is the adaptation of the second output Adaline from **State 1**. Assuming k adaptations have occurred to this point, the situation is as follows: $\vec{H}(k) = [+1, +1, -1]^T$, $o_2 = -1$, $d_2 = +1$, and $\vec{W}^{22}(k) = [-.15, .15, .15]^T$, where the bias input component is included in the hidden vector. For the moment, generalize the second Adaline's weight vector to $\vec{W}(k)^{22} = \alpha[-1, +1, +1]^T$. This generalized weight vector implements the same decision line but provides a generalized output confidence level controllable by the coefficient α.

The particular requirement is to adapt the second output Adaline to provide a $+1$ response. Note that if the weight update rule does not select a large enough weight adjustment to actually change the binary response of the second Adaline, the network remains in **State 1**. The next presentation of a pattern producing an error requires further adaptation

Postulated State 3					
input pattern		hidden pattern		output pattern	
x_1	x_2	h_1	h_2	o_1	o_2
+1	+1	−1	−1	+1	+1 *
+1	−1	+1	−1	−1	+1
−1	+1	+1	−1	−1	+1
−1	−1	+1	−1	−1	+1 *

Table 4.5: Mapping for the postulated **State 3**.

of the second output Adaline. This continues until **State 2** is finally reached. The weight update actually used by MRII will put the network in **State 2** in one step. The point here is that there is no advantage to adapting using smaller steps.

Upon reaching **State 2**, the weight vector for the second output Adaline has the form, $W(k+1)^{22} = [-\alpha + \delta(k), \alpha + \delta(k), \alpha - \delta(k)]^T$. A minimum value of $\delta(k)$ can be computed since in **State 2** it is required, $\vec{H}(k)^T \vec{W}(k+1)^{22} > 0$. The minimal value of $\delta(k)$ is $\alpha/3$.

For the adaptation from **State 1** to result in **State 3**, another minimal δ can be computed. The requirement to reach **State 3** is that the hidden pattern $h_1 = -1, h_2 = -1$, have response +1 by the second output Adaline. This requires $\delta(k) > 3\alpha$. Thus, to reach **State 3**, an adaptation step nine times larger than that needed to just reach **State 2** is required. This fact is independent of the value of α. Such steps may not be consistent with the minimal disturbance principle. Indeed, they may not be achievable by the particular weight update rule in use.

Consider what the response to $\vec{H}(k)$ is if **State 3** is reached. This value is what the adaptation level, L, would have been in the MRII weight update rule when adapting out of **State 1**. Setting $\delta(k) = 3\alpha$ in $\vec{W}^{22}(k+1)$ and taking the dot product with $\vec{H}(k)$ one gets 8α. For the example of this section, $\alpha = .15$. If **State 3** results by adapting from **State 1**, the response to the adapted on hidden pattern, $\vec{H}(k)$, is greater than 1.2.

The resulting confidence level is greater than the desired response for this hidden pattern. None of the weight update rules use step sizes that result in overshooting the desired response. Indeed, using such large steps is not in accordance with the minimal disturbance principle. MRII is based on the minimal disturbance principle. No weight update rule that is consistent with this principle will escape the limit cycle that results in this example.

4.3 Discussion

The previous section illustrates an example of MRII caught in a limit cycle. Why does this cycle happen? The reasons are twofold. At the outset, the first output Adaline reaches a solution before the second output Adaline. Second, the hidden pattern set settled upon by the first output Adaline is unusable by the second output Adaline in reaching a solution. Any attempt to change the hidden pattern set on the part of the second output Adaline is effectively vetoed by the first output Adaline. Any change to the hidden pattern set destroys the solution arrived at by the first output Adaline. Thus, the system enters a local error minimum. No change on the first layer can reduce output errors.

The second layer Adalines are in a situation not unlike that which inspired the usage concept. There is a nonseparable subset of hidden patterns that the second output Adaline is forced to try to separate. This Adaline is successful at performing the separation for a given presentation but cannot come to a global solution. Unfortunately, usage cannot be employed at the output layer. When the output layer is forced to adapt, there is no freedom left to share responsibility for correcting an output.

At this point, it is important to note that usage, even on the first layer, cannot resolve this particular problem. In examining the possible trial adaptations from **State 1**, all are considered and rejected. Usage affects which Adalines participate in trial adaptations when participation is limited to some number less than the total in the layer. It can have no effect when all Adalines are considered for participation.

In this example, the limit cycle is easily recognizable. It is characterized by a lack of accepted trial adaptations on the first layer, and by a single output Adaline being adapted over and over. The master controller could detect a lack of accepted trial adaptations. In this case, there is hope for implementing an escape mechanism since a developed limit cycle can be recognized.

For larger networks with larger training sets, the failure mode may not be so easily recognized. It may be possible for some output units to come to solutions on subsets of the training set. These units would then limit the allowable hidden patterns to a small set. This small set would not allow the other units to come to a global solution but still allow some first-layer trial adaptations to be accepted. The limit cycle would consist of a large number of specific states. By restricting the hidden patterns to a nonsolvable set, wide variations of training performance could be seen. This would effectively mask the failure mode. The author conjectures situations like this occur in some of the failures to converge

with the emulator problem of Section 3.4.

Though not reported in the previous chapter, MRII solves the n-bit parity problem. The network required is an n-input, n-feed-1 Madaline. The task is to classify the input pattern as $+1$ if it has an even number of $+1$s among its n variable input components and -1 otherwise. Training is generally successful but a failure mode similar to the one noted above is observed. The parity problem has a single output, so there is no possibility of another output Adaline vetoing trial adaptations as with the and/xor problem. What happens instead is that the allowed trial adaptations are ineffective. The ones tried do not correct the output. This happens because of the somewhat arbitrary limit placed on the number of Adalines that will be considered for adaptation. The generally useful heuristic of considering the least confident half of the Adalines in a given layer sometimes fails. In this case, it is easy for the master controller to increase the number of Adalines considered for adaptation and increase the number of Adalines that are trial adapted at one time. Unfortunately, this procedure is very heuristic. It does, however, work. With some experience, methods using this concept can be implemented to insure training on the n-bit parity problem almost always converges. The fine tuning used to achieve good convergence for the n-bit parity problem does not always transfer to another problem, nor even to the $(n+2)$-bit parity problem, so the details are omitted here.

Sometimes with smaller nets, *all* possible trial adaptations are rejected. The cause for this is a large initial bias weight randomly chosen for the output Adaline. The bias determines the network's response, and the first-layer Adalines aren't weighted sufficiently to have any effect. A simple solution to this is to initialize the weights in the output Adaline to all have equal values. This Adaline then starts as a majority voter, giving every first-layer Adaline equal say in the network response. In any event, these techniques do not allow escape from the and/xor limit cycle detailed above.

The specific weights for the and/xor example cited in this chapter result after several adaptations are made. Starting with the same initial weights that led to the limit cycle, but using a different pattern presentation sequence, MRII avoids the limit cycle and instead converges to a solution quite quickly. The point is, there seems to be no way to characterize a set of initial weights as being a set that leads to a limit cycle. Indeed, a cursory look at the weights presented at the beginning of the chapter shows nothing remarkable about such weights. They could easily have occurred at initialization. Indeed, weights that define a limit cycle condition sometimes do occur at initialization. It seems unlikely that a method for avoiding limit cycles can be found while retaining the basic form of the algorithm. The

only feasible strategy seems to be to detect the limit cycle and then implement some escape.

The minimal disturbance principle underlies the MRII algorithm. Any limit cycle escape mechanism will have to selectively depart from the minimal disturbance principle, perhaps in an extreme fashion. The size of the adaptation level needed to reach **State 3** is an example of how extreme this departure may have to be.

Various techniques have been used with varying degrees of success. Most are of a heuristic nature, and often need to be hand crafted for the particular problem being solved. Some of these techniques include: allowing the adaptation level of non-output units to be a random variable; modifying the output unit adaptation levels by their usage count; and modifying the number of Adalines allowed to participate in trial adaptations by the average output unit usage count. None of these offer much hope of being generally applicable.

Probably the easiest escape, that is generally applicable, is to reinitialize the weights in the system and start the training over. Often this is not a bad thing to do. Experience indicates that if MRII is going to find a solution it will find it fairly quickly. If training performance levels off at an unacceptable level, after a reasonable time, start over. The determination of a reasonable time can be done by doing an ensemble of training attempts. A look at the average learning curve will reveal an average performance increase versus the learning opportunities allowed. Extrapolate the curve to the projected convergence point. Use this point to set a reasonable limit to the number of learning opportunities to allow before reinitializing.

Reinitialization has the drawback of losing all benefit of training performed up to that point. Often the system will be somewhere near a solution and merely needs to be nudged out of a local minimum. One way to do this is to occasionally add some random perturbations to the weights. This moves the system at random to a nearby place in weight space. The idea is to get away from the local minimum but not so far away from the previous weight position as to negate all previous training.

This weight perturbation method is expected to be generally applicable. Some preliminary success was achieved using this technique with the n-bit parity problem. The required perturbation magnitudes and a schedule for applying them were determined by experience. The magnitude of perturbations required for a general problem may depend only on the particular architecture being trained. Thus, an analysis of the effect of weight perturbations on the input/output mapping of a Madaline system could be useful in generalizing this technique. This reasoning is one of the prime motivations for the sensitivity work presented in the next chapter.

Chapter 5

Sensitivity of Madalines to Weight Disturbances

5.1 Motivation

The training performance of Madaline Rule II rises quickly and then levels off, as experimental results have shown. For even moderately sized networks, the number of training attempts resulting in convergence represents a smaller fraction of the total attempts than one would like. For some architectures, considerable refinement of the number of Adalines allowed to participate in trial adaptations and adjustment of the "MULT" factor of the usage gain is required to get performance levels like those reported in Chapter 3. The identified limit-cycle failure mode, presented in the previous chapter, may account for many of the failures to converge. In larger networks, this failure mode is difficult to recognize. There may be a more fundamental issue, however.

For the emulator problem of Section 3.4, the weights in the fixed net are generated randomly. As such, the mapping from input patterns to outputs may change significantly for small changes in the weights of the fixed net. If this is the case, how close do the weights in the trained adaptive net have to be to the weights of the fixed net to get close to identical performance? What is the rate of difference error in the outputs of the two networks as the weights of the adaptive network deviate further and further from the weights of the fixed network?

An investigation of the sensitivity of the input/output mapping to changes in the weights may lead to insightful improvements of the MRII algorithm. It was proposed at the end of

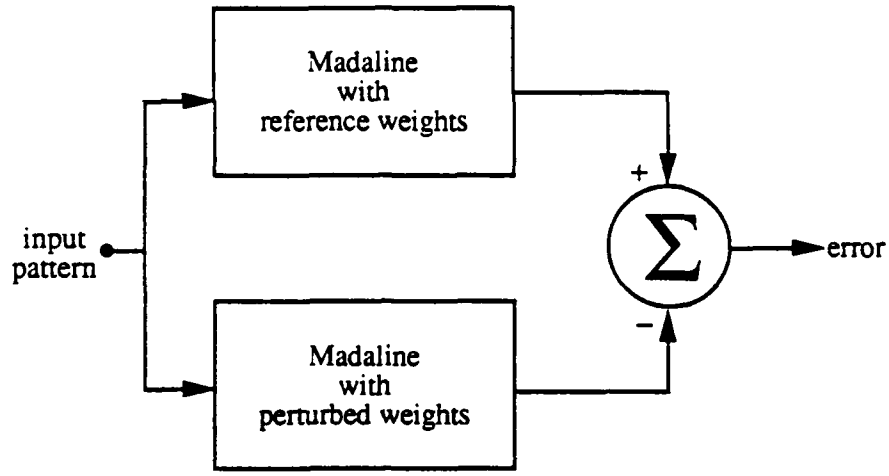


Figure 5.1: Comparing a network with a weights perturbed version of itself.

the last chapter to use random weight perturbations to escape from the algorithm's failure mode. A thoughtful approach to this method requires gaining some understanding of the weight sensitivity issue. The study of sensitivity may also suggest modifications of the network architectures and input representations in order to minimize the effects of weight disturbances. Knowledge of sensitivity also gives one some idea of how precisely the weights of an adaptive network need to be set in order to achieve precise output-response objectives. This is an important consideration when adaptive hardware is analog rather than digital. It is also an important issue when considering how long to run an adaptive process to get closer and closer to perfection.

The investigation of sensitivity uses a structure similar to that of Figure 3.4. Instead of an adaptive network being compared to a fixed network, a perturbed version of a net is compared against the unperturbed net, called the reference net, as shown in Figure 5.1. The analyses use the Hoff hypersphere approximation which is presented next.

5.2 Hoff Hypersphere Approximation

Hoff [12] is responsible for the hypersphere approximation concept that allowed most of the analytical results on Adalines and Madalines to date to be derived. Hoff formulated this approximation in the n-dimensional pattern space. This section explains the hypersphere approximation and introduces some facts about hypersphere geometry.

In three-dimensional pattern space it is easy to visualize the locations of all the binary

input patterns as being the vertices of a cube. For spaces with higher dimensionality, the patterns are found at the vertices of a hypercube. The input patterns all have the same magnitude, $\sqrt{n}$. Thus, the hypercube can be inscribed in a hypersphere of radius $\sqrt{n}$. The vertices of a hypercube have a regular arrangement symmetric about the origin. Hoff postulated that as n gets large, one could say the input patterns are uniformly distributed on the surface of the hypersphere. This immerses the case of binary input patterns into the continuous analog patterns case. He showed this was a valid assumption in his doctoral dissertation.

The power of this assumption is that it allows the use of probabilistic methods to analyze the Adaline. The probability of an input pattern lying in a particular region of the input space could be computed as the ratio of the area of the region on the hypersphere to the area of the whole hypersphere. Thus, instead of performing summations over discrete points, integrals over regions of the hypersphere could be used in many of the analyses that needed to be done. One use of these techniques was to prove that the capacity of an Adaline to store random patterns was equal to twice the number of weights in the Adaline [13].

Glanz [3] applied the hypersphere approximation to the $(n+1)$-dimensional input vector space. He argued that since the x_0 component could only take on half the values a variable component could, the hypersphere approximation would apply to half the hypersphere. The input vectors can be thought of as being uniformly distributed on the hemihypersphere in the positive x_0 half-space.

Most previous analyses were performed assuming a unit hypersphere. In these analyses, only the directions of the input and weight vectors were important. In the analytical results presented in this work, the magnitudes of vectors are important and care must be taken to work with the proper sized hypersphere.

At this a point, a few facts about hyperspheres are introduced. Further information can be found in Kendall [14] or Sommerville [15].

- Strictly speaking, the area of a hypersphere should be called its surface content but the term area will usually be used. The area of a hypersphere of radius r in n-dimensional space is given by:

$$A_n = K_n r^{n-1} \tag{5.1}$$

where,

$$K_n = \frac{2\pi^{n/2}}{\Gamma(n/2)}.$$

The expression for K_n can be written in terms of factorials instead of the gamma function if distinction is made for n even and odd:

$$K_n = \frac{2\pi^m}{(m-1)!} \qquad \text{for } n = 2m \tag{5.2}$$

$$= \frac{2^{2m+1}\pi^m m!}{(2m)!} \qquad \text{for } n = 2m+1 \tag{5.3}$$

- The intersection of a hyperplane and a hypersphere in n-space is a hypersphere in $(n-1)$-space. For $n = 3$, this says a plane intersects a sphere in a circle. The center of the reduced dimension hypersphere is the projection of the center of the n-sphere onto the intersecting hyperplane.

- The differential element of area, dA_n, on a hypersphere of radius r, as a function of one polar angle ϕ is given by:

$$dA_n = K_{n-1}\, r^{n-1} \sin^{n-2}\phi \, d\phi \tag{5.4}$$

  This can be understood by realizing that a particular value of ϕ defines an $n-1$ dimensional hypersphere of radius $r \sin\phi$. The differential area element is the surface content of this hypersphere of reduced dimension multiplied by a thickness $r\, d\phi$. A representation of this in three dimensions is shown in Figure 5.2.

- The hyperplane containing the origin and perpendicular to a vector emanating from the origin will divide a hypersphere centered on the origin into two hemihyperspheres. In similar fashion, a second vector will define a hyperplane that bisects the hypersphere. The surface contained between specific sides of two such hyperplanes is called a lune (see Figure 5.3). If the angle between the two vectors is θ, the surface content of the lune is given by:

$$\text{Area of lune} = \frac{\theta}{2\pi} A_n \tag{5.5}$$

With this information as background material, an analysis of the sensitivity of the single Adaline to changes in its weights can be done.

5.3 Perturbing the Weights of a Single Adaline

Hoff [12] did an analysis of the effect of weight perturbations on the decision mapping of an Adaline using the hypersphere approximation. He did his analysis in the n-dimensional

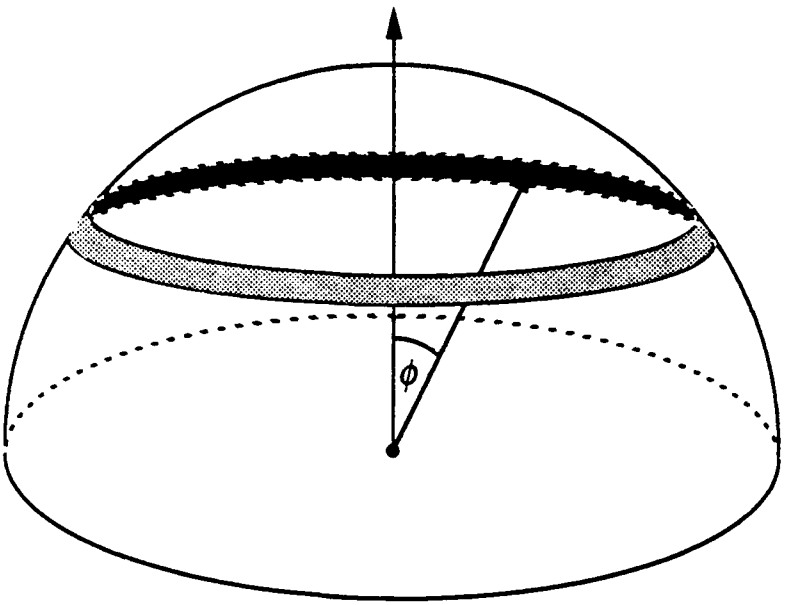


Figure 5.2: The differential area element of a hypersphere as a function of the polar angle ϕ.

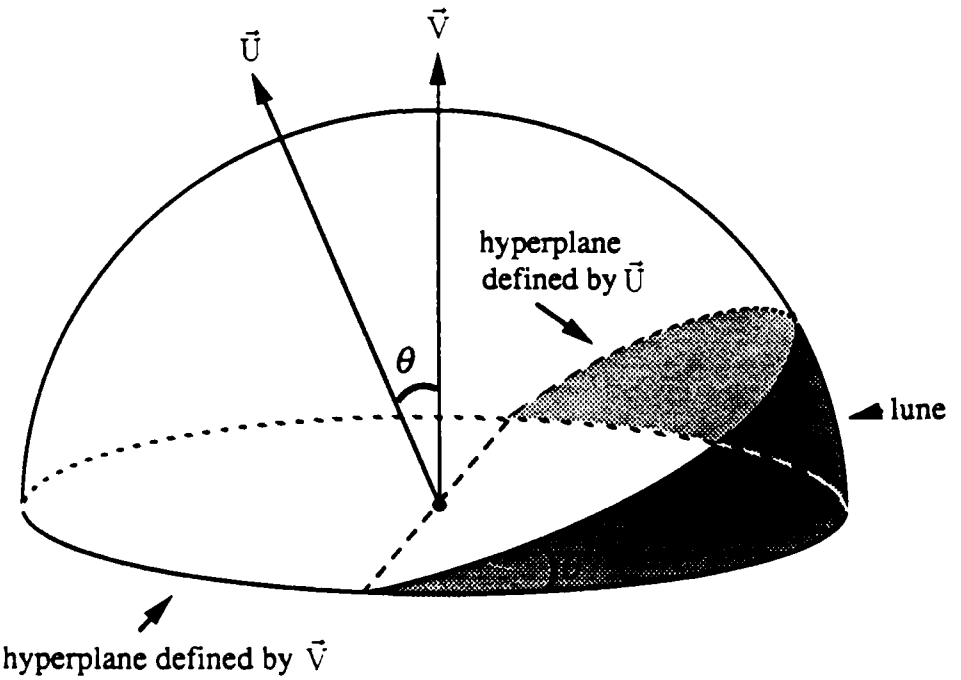


Figure 5.3: A lune of angle θ formed by two bisecting hyperplanes.

pattern space. The bias weight introduces a complication to such an analysis. The bias weight is not considered part of the weight vector in n-space analyses. This causes the separating hyperplane to pass through the pattern hypersphere offset from the origin by a distance dependent on the bias weight, and the magnitude of the n-dimensional weight vector. Changes in the weights must be resolved into components parallel and perpendicular to the weight vector. The final results were complicated and difficult to apply in answering the question of how close weights have to be to a known solution for good performance.

By using the hypersphere approximation in the $(n+1)$-dimensional weight space, simplifications occur. The input pattern is now considered to have dimension $n+1$, but with the bias input, x_0, a constant $+1$. Glanz's modification of the hypersphere approximation can be used. The patterns are now confined to an oriented half of the hypersphere but within this region they are assumed uniformly distributed. The major simplification is that now the separating hyperplane always passes through the origin. The decision of which input patterns are classified $+1$ and which are -1 is determined by how the hyperplane intersects the pattern hemihypersphere. Changes in the weights cause the orientation of the hyperplane to change. These changes can be described by a single parameter, the angle between the original weight vector and the new weight vector.

In Figure 5.4, the hemihypersphere corresponding to a positive x_0 is shown, as are a weight vector, a perturbed weight vector, and the intersections of the associated decision hyperplanes. The result of perturbing the weight vector is a reorientation of the decision hyperplane. For those patterns in the darker shaded region, the perturbed network makes a "decision error" relative to the output response of the reference network of Figure 5.1.

The darker shaded area consists of two partial lunes. By symmetry, the area of the two partial lunes add to that of a lune of angle θ, the angle between the weight and perturbed weight vectors. Using the hypersphere approximation, the probability of a decision error due to a shift in the weight vector is:

$$\begin{aligned} \text{P[Decision Error]} &= \frac{\text{Area of shaded lune}}{\text{Area of hemihypersphere}} \\ &= \frac{\theta}{\pi} \end{aligned}$$

Taking the expectation of both sides,

$$\text{Ave P[Decision Errors]} = \frac{\text{E}\{\theta\}}{\pi} = \frac{\overline{\theta}}{\pi} \tag{5.6}$$

To determine $\overline{\theta}$, refer to Figure 5.5. In $(n+1)$-space, $\vec{W}$ and $\Delta\vec{W}$ determine a plane

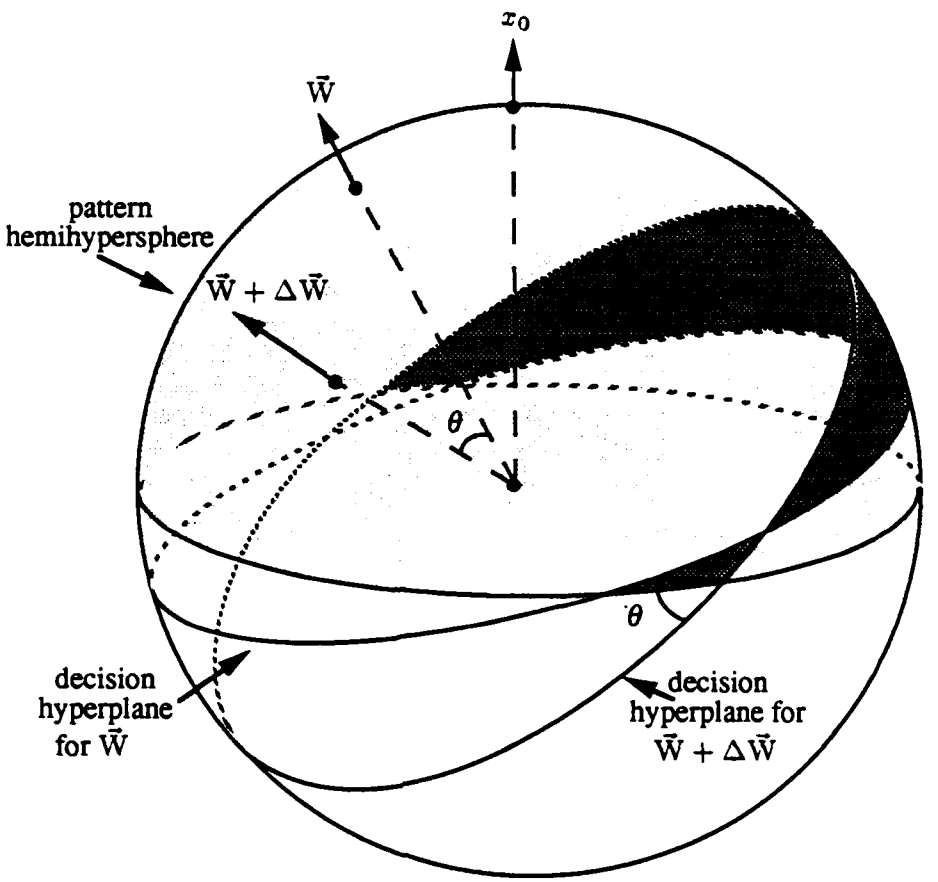


Figure 5.4: Reorientation of the decision hyperplane due to a disturbance of the weight vector. Patterns lying in the darker shaded region change classification.

and the angle between them in that plane, ϕ. The angle, θ, between $\vec{W}$ and $\vec{W} + \Delta\vec{W}$ is found using plane geometry.

$$\begin{aligned}
\theta &= \tan^{-1} \frac{|\Delta\vec{W}| \sin\phi}{|\vec{W}| + |\Delta\vec{W}| \cos\phi} \\
&= \tan^{-1} \frac{\sin\phi}{\frac{|\vec{W}|}{|\Delta\vec{W}|} + \cos\phi} \\
&\approx \tan^{-1} \left(\frac{|\Delta\vec{W}|}{|\vec{W}|} \sin\phi \right) \quad \text{for } \frac{|\vec{W}|}{|\Delta\vec{W}|} \gg 1
\end{aligned}$$

Using $\tan^{-1} x \approx x$ for small x,

$$\theta \approx \frac{|\Delta\vec{W}|}{|\vec{W}|} \sin\phi$$

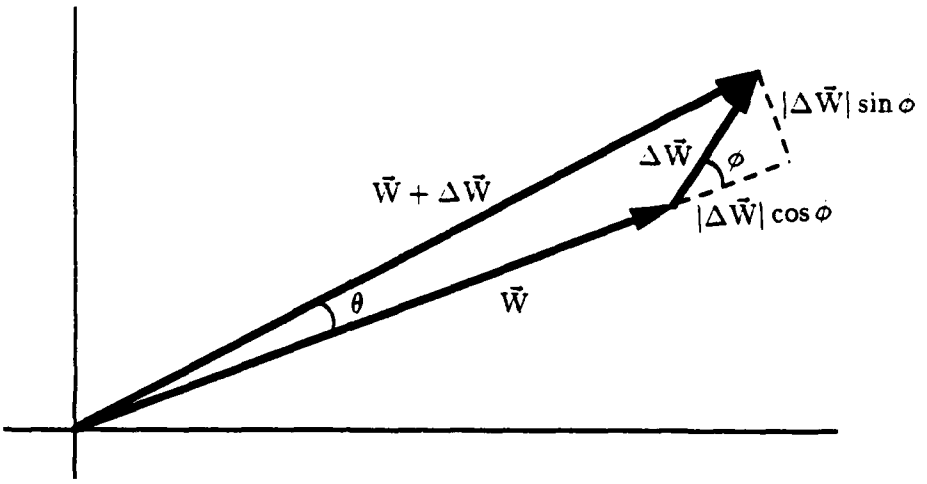


Figure 5.5: Geometry in the plane determined by $\vec{W}$ and $\Delta\vec{W}$.

Then,

$$\overline{\theta} \approx \frac{|\Delta\vec{W}|}{|\vec{W}|} E\{\sin\phi\}$$

It's assumed here that the ratio of the magnitude of the weight perturbation vector to the magnitude of the original weight vector is known and constant. If this is not the case, but the perturbations are independent of each other and independent of the original weights, an average perturbation magnitude can be used.

To complete the derivation of the average decision errors a single Adaline makes due to perturbation of its weights, a formula for $E\{\sin\phi\}$ is needed. The vectors $\vec{W}$ and $\Delta\vec{W}$, or their extensions, will intersect a hypersphere of unit radius. Since only the directions of these vectors determine ϕ, the analysis can assume a unit radius sphere without loss of generality. Glanz [3] notes that the probability that the angle between two vectors, $\vec{V}_1$ and $\vec{V}_2$, has value between ϕ and $\phi + \Delta\phi$ is proportional to the differential area at angle ϕ (see Equation 5.4). Modifying his equation for use in $(n+1)$-space, the probability density function can be written,

$$p[\ \vec{V}_1 \text{ and } \vec{V}_2 \text{ form angle } \phi\ ] = \frac{K_n}{K_{n+1}} \sin^{n-1}\phi \tag{5.7}$$

Therefore,

$$E\{\sin\phi\} = \int_0^\pi \sin\xi \frac{K_n}{K_{n+1}} \sin^{n-1}\xi \, d\xi$$

$$
\begin{aligned}
&= \frac{2K_n}{K_{n+1}} \int_0^{\frac{\pi}{2}} \sin^n \xi \, d\xi \\
&= \frac{2K_n}{K_{n+1}} \frac{\sqrt{\pi}}{2} \frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}+1\right)} \\
&= \frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} \frac{\Gamma\left(\frac{n+1}{2}\right)}{2\pi^{\frac{n+1}{2}}} \sqrt{\pi} \frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}+1\right)} \\
&= \frac{\Gamma^2\left(\frac{n+1}{2}\right)}{\Gamma^2\left(\frac{n}{2}\right)} \frac{2}{n} \\
&= \left(\frac{K_n}{K_{n+1}}\right)^2 \frac{2\pi}{n} \qquad (5.8)
\end{aligned}
$$

This result is further simplified by using Stirling's approximation for factorials and the expressions for K_n for n even and odd in Equations 5.2 and 5.3. Stirling's approximation is:

$$n! \approx \sqrt{2\pi n}\, n^n e^{-n}.$$

For $n = 2m$,

$$
\begin{aligned}
\frac{K_n}{K_{n+1}} &= \frac{2\pi^m}{(m-1)!} \frac{(2m)!}{\pi^m 2^{2m+1} m!} \\
&= \frac{(2m)!}{2^{2m}\, m!\,(m-1)!} \\
&= \frac{m\,(2m)!}{2^{2m}\,(m!)^2} \\
&\approx \frac{m\sqrt{4\pi m}\,(2m)^{2m} e^{-2m}}{2^{2m}\, 2\pi m\, m^{2m} e^{-2m}} \\
&\approx \sqrt{\frac{m}{\pi}} = \sqrt{\frac{n}{2\pi}} \qquad \text{for } n \text{ even.}
\end{aligned}
$$

For n odd, $n = 2m + 1$ and $n + 1 = 2(m + 1)$,

$$
\begin{aligned}
\frac{K_n}{K_{n+1}} &= \frac{\pi^m 2^{2m+1}(m)!}{(2m)!} \frac{m!}{2\pi^{m+1}} \\
&= \frac{2^{2m}\,(m!)^2}{\pi\,(2m)!} \\
&\approx \frac{2^{2m}\left(\sqrt{2\pi m}\, m^m e^{-m}\right)^2}{\pi\sqrt{2\pi 2m}\,(2m)^{2m} e^{-2m}}
\end{aligned}
$$

$$\approx \frac{2\pi m}{\pi\sqrt{4\pi m}}$$

$$\approx \sqrt{\frac{m}{\pi}} = \sqrt{\frac{n-1}{2\pi}} \qquad \text{for } n \text{ odd.}$$

For n sufficiently large, the even and odd cases become indistinguishable and it can be approximated,

$$\frac{K_n}{K_{n+1}} \approx \sqrt{\frac{n}{2\pi}} \qquad \text{for } n \text{ large} \tag{5.9}$$

With this result, Equation 5.8 becomes,

$$\begin{aligned} E\{\sin\phi\} &\approx \left(\sqrt{\frac{n}{2\pi}}\right)^2 \frac{2\pi}{n} \\ &\approx 1 \qquad \text{for } n \text{ large} \end{aligned}$$

The final result for determining the change in the input/output mapping of an Adaline caused by a disturbance of its weights can now be formulated. Using the results so far,

$$\begin{aligned} P[\text{Decision Error}] &= \frac{\theta}{\pi} \\ &\approx \frac{1}{\pi}\frac{|\Delta\vec{W}|}{|\vec{W}|}\left(\frac{K_n}{K_{n+1}}\right)^2 \frac{2\pi}{n} \qquad (5.10) \\ &\approx \frac{1}{\pi}\frac{|\Delta\vec{W}|}{|\vec{W}|} \qquad (5.11) \end{aligned}$$

The last two expressions represent different levels of approximation. Future reference will call the first the complicated approximation while the second will be called the simple approximation. The simple approximation ignores any dependency on the number of inputs and is good in the limit as n gets large. It tells us that the percentage of decision errors, on average, is equal to the average percent error in the Adaline's weights multiplied by $1/\pi$.

5.4 Decision Errors Due to Input Errors

The above result can be used to predict the error rate of an Adaline in the first layer of a multi-layer network. Let the weights of the fixed reference network be randomly chosen. Let the weights of the adaptive network be set to corresponding values except for small random errors. As such, it can be assumed that the first-layer Adalines of this perturbed network will have independent output errors. Assume that all Adalines of this first layer

are affected by weight disturbances such that their weight perturbation ratios, $|\Delta\vec{W}|/|\vec{W}|$, are, on average, the same. They will all have, therefore, the same probability of making an error. Let this probability of error be designated $\mathcal{E}$ and the number of first-layer Adalines be n_1. The output of the first layer of a Madaline is called the first hidden pattern. What is the probability that the first hidden pattern of the perturbed network has h errors in it relative to the first hidden pattern of the reference net? Due to the assumed independence of the first-layer Adalines of the perturbed net and the assumed equal error rate for them, this probability can be written [16]:

$$\text{P[hidden pattern has } h \text{ errors]} = \frac{n_1!}{h!(n_1-h)!}\mathcal{E}^h(1-\mathcal{E})^{n_1-h} \tag{5.12}$$

The number of errors in the output of this first layer of a perturbed network relative to the reference network is binomially distributed. This distribution has as parameters the number of first-layer Adalines and the error rate of an Adaline.

The perturbed network's second layer will see inputs that are erroneous relative to those seen by the second layer of the reference network. Some of these erroneous patterns will cause decision errors at the network output even if there are no weight errors in the second and subsequent layers. The basic question to investigate then is, how do random input errors affect an Adaline's decision mapping?

In the binary case where the components of the input are either $+1$ or -1, an input error can be thought of as a "flipped" bit. If a pattern has one flipped bit, the result is a pattern of Hamming distance one from the original. One way to think of flipping a bit is to add a $\Delta\vec{X}$ to $\vec{X}$. To generate a nearest neighbor of $\vec{X}$, add a vector to it that has all components equal to zero except for one, say the ith. Let the ith component, Δx_i be equal to $-2x_i$. This disturbance vector has a Euclidean length of two.

In general, a pattern vector which is a Hamming distance h from a given pattern $\vec{X}$ can be generated by flipping h of its bits. This is done by adding a disturbance vector $\Delta\vec{X}$ having h nonzero components equal to -2 times the corresponding $\vec{X}$ component. This $\Delta\vec{X}$ has a Euclidean length of $\sqrt{4h}$.

Note that all of $\vec{X}$'s pattern neighbors of Hamming distance h are at the same Euclidean distance from $\vec{X}$. Thus, the pattern neighbors of given Hamming distance from $\vec{X}$ lie on a hypersphere of radius $\sqrt{4h}$ with the tip of $\vec{X}$ as origin. The neighbors also lie on the pattern hemihypersphere on which $\vec{X}$ is located, since they are legitimate patterns in the input space. The possibility of the bias input being flipped is excluded. The intersection of these two hyperspheric regions is a hypersphere of dimension one less than the pattern

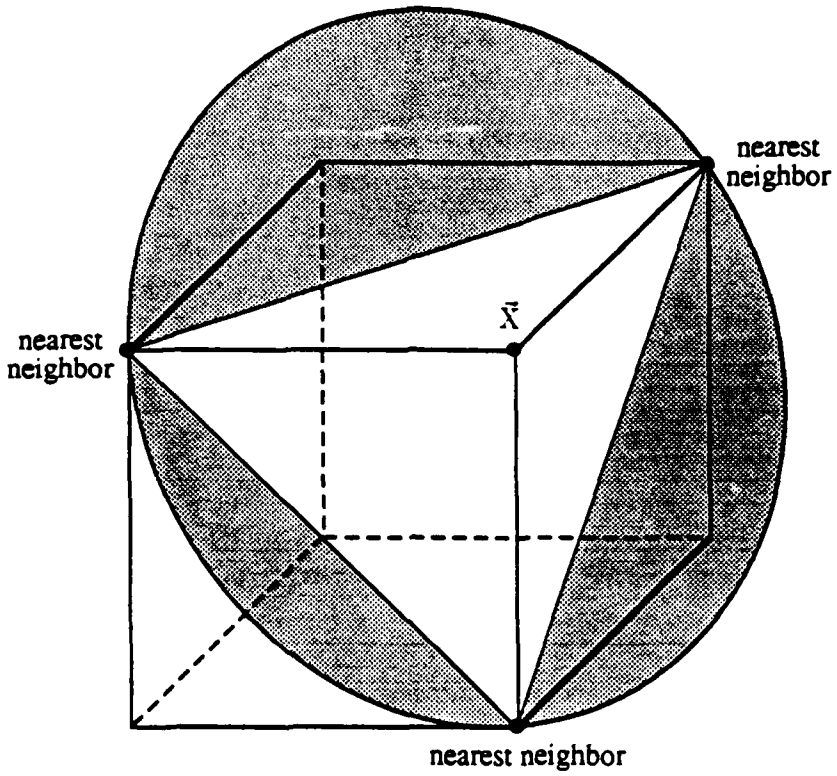


Figure 5.6: The nearest neighbors of $\vec{X}$ lie equally spaced on a reduced-dimension hypersphere.

hemihypersphere. Hoff's hypersphere approximation leads one to conjecture that the pattern neighbors of $\vec{X}$ are uniformly distributed on this surface of reduced dimension. A visualization that supports this conjecture is shown in Figure 5.6. Consider the situation where $n = 3$ and there is no bias input. The eight patterns available in 3-space form the vertices of a cube which can be inscribed in a sphere (not shown in the figure). The vertex marked $\vec{X}$ has three nearest neighbors. They lie equally spaced on a circle which lies in a plane. Furthermore, the vector $\vec{X}$ intersects this plane at the center of the circle. For the following analysis, it will be assumed that the conjecture of a pattern's neighbors being uniformly distributed in the reduced-dimensional space is true.

If the weight vector components are chosen from a zero mean distribution, the weight vector will be orthogonal to the x_0 axis, on average. This is because the expected value of the dot product of the weight vector with the x_0 axis is the expected value of the bias weight, which is zero since it was selected from a zero mean distribution. This leads to the intuitive conclusion that an "average" Adaline will have equally likely +1 and -1 outputs.

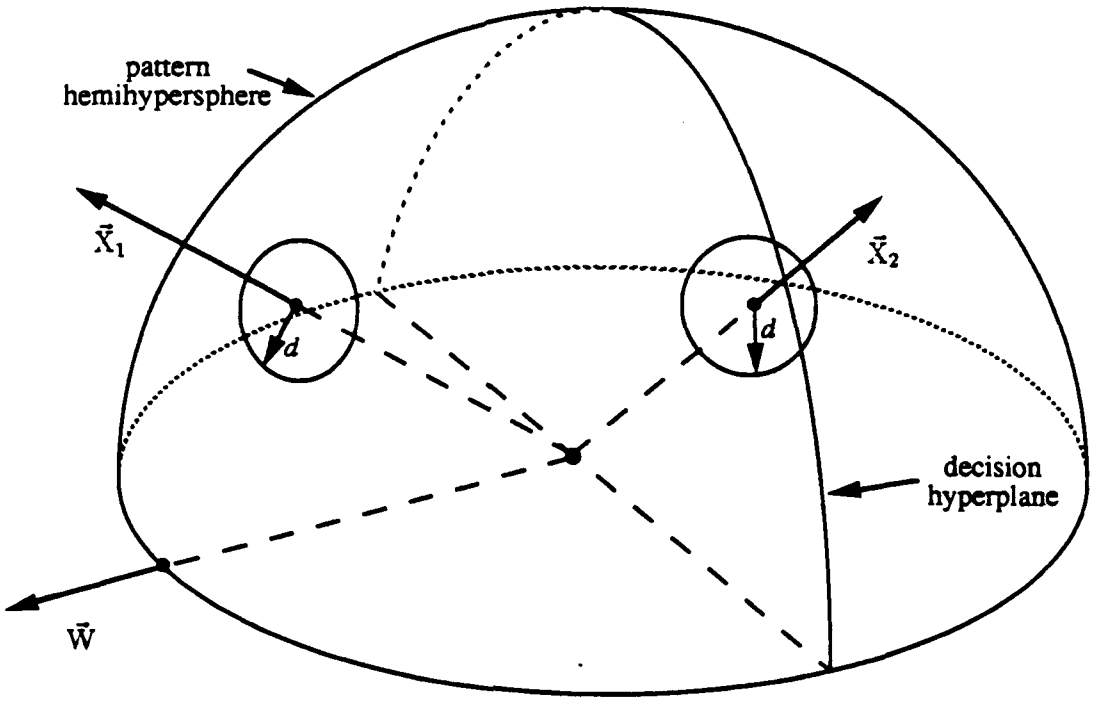


Figure 5.7: All the patterns at distance d from $\vec{X}_1$ will be classified the same as $\vec{X}_1$. Some of the patterns at distance d from $\vec{X}_2$ are classified differently from $\vec{X}_2$.

This average Adaline will be analyzed to determine the effect of random input errors on its input/output map.

The analysis is aided by a study of the drawings of Figure 5.7. Without loss of generality, assume the pattern vector $\vec{X}$ of interest lies on the positive side of the decision hyperplane. Shown in the figure is a representation of the location of the pattern neighbors which are located Euclidean distance d from two different $\vec{X}$s. $\vec{X}_1$ is located sufficiently far from the decision hyperplane that all of its pattern neighbors of distance d away will be classified positive. No errors will be made by the Adaline if any of these neighbors are presented instead of the correct pattern $\vec{X}_1$. The pattern $\vec{X}_2$ is located closer to the decision hyperplane and some of its neighbors at distance d are on the opposite side of the decision hyperplane. Thus, in a fraction of those instances that a neighbor is presented instead of the correct $\vec{X}_2$, a decision error will be made by the Adaline. It will be determined here what fraction of these presentations of pattern neighbors will result in decision errors.

At this point, the notions of dimensionality need to be made more precise. The terminology employed by Sommerville [15] is used. The input patterns lie on a hypersphere of n

dimensions in $(n+1)$-space. The hypersphere is of dimension n because only n components are needed to specify a point on this surface. The 2^n patterns that comprise the input space of the Adaline are confined to the positive x_0 hemihypersphere. The bias input is supplied internally by the Adaline and is considered not subject to error. The decision hyperplane of an Adaline, described analytically by $\vec{X}^T\vec{W} = 0$, is a surface of n dimensions. Sommerville calls this an n-flat but it will be referred to as an n-dimensional hyperplane here. The pattern neighbors of $\vec{X}$ at Euclidean distance d are located on a surface defined by the intersection of two hyperspheres of n dimensions in $(n+1)$-space. The first hypersphere is the one of radius $\sqrt{n+1}$ centered at the origin and the second is the hypersphere of radius $d = \sqrt{4h}$ centered about the tip of $\vec{X}$. This intersection is a hypersphere of dimension $n-1$. Furthermore, this intersection lies in an n-dimensional hyperplane. This is seen as follows. Let $\vec{N}$ be a pattern neighbor. It is representable by $\vec{N} = \vec{X} + \Delta\vec{X}$ where $\Delta\vec{X}$ is of the form described earlier. Then,

$$\begin{aligned}\vec{X}^T\vec{N} &= \vec{X}^T(\vec{X} + \Delta\vec{X}) \\ &= \vec{X}^T\vec{X} + \vec{X}^T\Delta\vec{X} \\ &= n + 1 + -2h = \text{a constant},\end{aligned}$$

h being the Hamming distance of the neighbor. This is the equation of an n-dimensional hyperplane at a distance from the origin of $(n+1-2h)/\sqrt{n+1} = r - (2h/r)$ where $r = \sqrt{n+1}$ is the radius of the pattern hypersphere. Thus, the pattern neighbors of $\vec{X}$ at Euclidean distance d lie on a hypersphere of dimension $n-1$ which is contained in an n-dimensional hyperplane.

Visualization of the details of the geometric analysis is aided by the drawings in Figures 5.8–5.10. Figure 5.8 shows a representation of the situation in the $(n+1)$-dimensional pattern/weight space. Here, the perspective is rotated so that the true pattern vector $\vec{X}$ is pointing up. The pattern neighbors of $\vec{X}$ that lie at Euclidean distance d from $\vec{X}$ are represented by a circle which lies on the pattern sphere. This circle lies in a plane which cuts off a spherical cap. This cap is shaded in the figure to aid visualization. More precisely, the patterns are located on a hypersphere of n dimensions. The pattern neighbors are on a hypersphere of $n-1$ dimensions which lies in an n-flat which cuts off a hyperspherical cap from the pattern hypersphere. Because of the perspective, the hemisphere shown in the figure no longer represents the pattern hemihypersphere. The pattern neighbors of $\vec{X}$ do lie on the pattern hemihypersphere, however, and it will be assumed that their distribution is

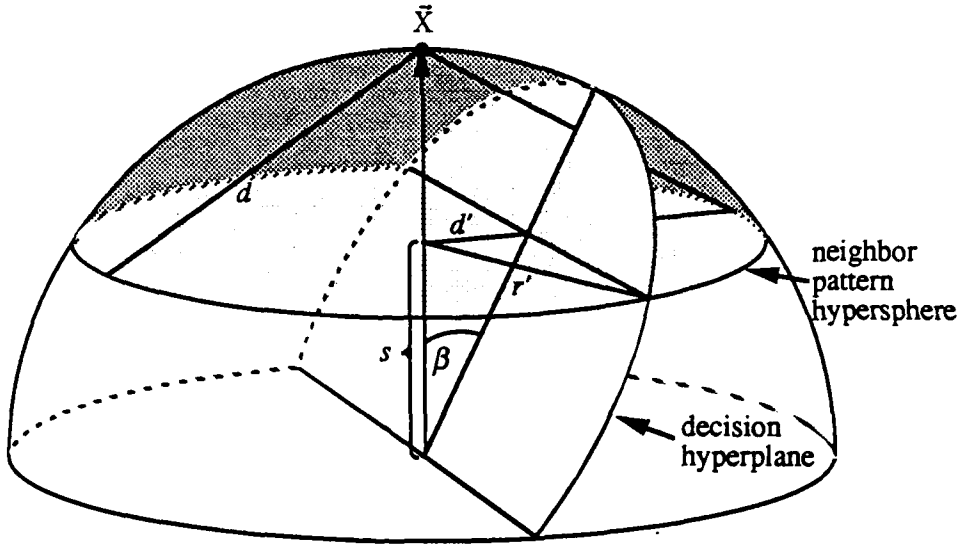


Figure 5.8: A representation of the location of the patterns at distance d from the input vector $\vec{X}$.

unaffected by "edge" effects caused by the actual position of the pattern hemihypersphere. This assumption is good for the average Adaline that has only a small number of errors in the input pattern presented.

The pattern vector $\vec{X}$ and weight vector $\vec{W}$ define a plane. Figure 5.9 shows the geometry in this plane. This plane intersects the decision hyperplane in a line, and the pattern hypersphere in a circular arc. The plane intersects the hypersphere of $n-1$ dimensions containing the pattern neighbors in two points. Ordinary plane geometry can be used for analysis in this plane formed by $\vec{X}$ and $\vec{W}$.

A representation of the hyperplane containing the pattern neighbors is shown in Figure 5.10. This hyperplane forms the floor of the hyperspherical cap, and the circle represents the hypersphere in which the pattern neighbors lie. The center of this hypersphere is the intersection of the hyperplane containing the neighbor patterns and the vector $\vec{X}$. The decision hyperplane intersects the hyperplane containing the pattern neighbors in a hyperplane of dimension $n-1$ or an $(n-1)$-flat, represented by a line in the figure. The pattern neighbors on the opposite side of the decision hyperplane from the center of the pattern neighbor hypersphere will cause errors if they are presented to the Adaline instead of the true pattern $\vec{X}$. The Hoff hypersphere approximation says the probability of error will be the ratio of the area of the neighbor hypersphere on the opposite side of the decision hyperplane to the

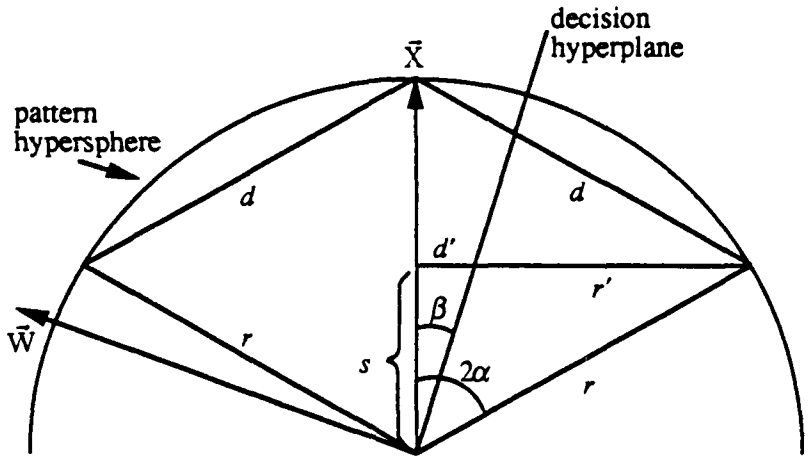


Figure 5.9: A representation of the geometry in the plane formed by $\vec{W}$ and $\vec{X}$.

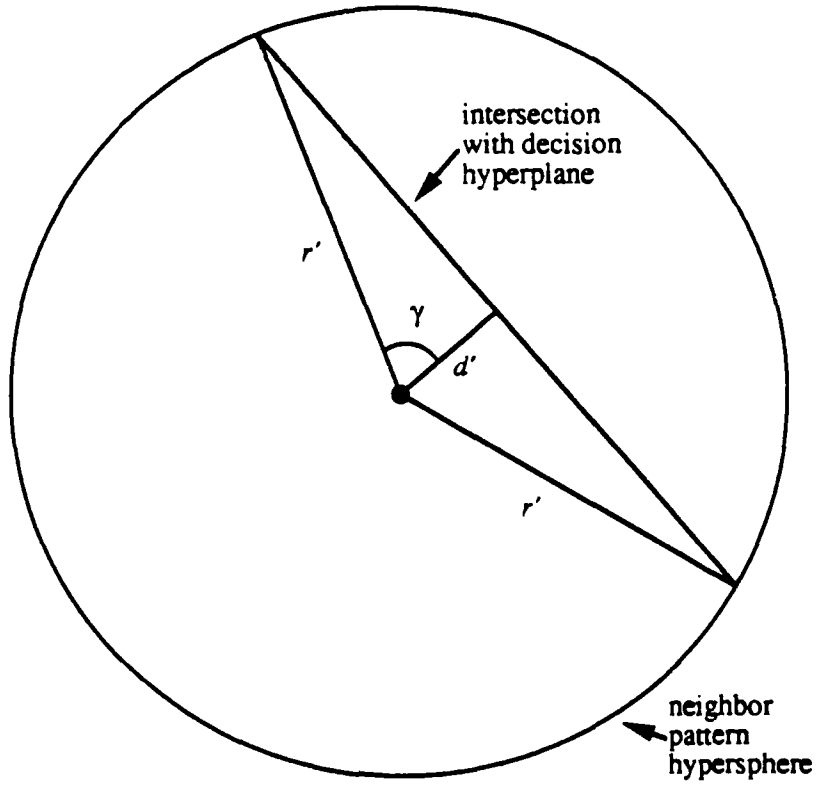


Figure 5.10: Representation of the geometry in the hyperplane containing the patterns at distance d from $\vec{X}$. This hyperplane is the floor of the shaded cap in Figure 5.8.

total area of the neighbor hypersphere. The radius of the neighbor pattern hypersphere is designated r'. The distance from the center of this hypersphere to the intersection with the decision hyperplane is designated d'. Determining r' and d' allows the determination of the angle γ, which can be used to compute the required area on the opposite side of the decision hyperplane. The quantities r' and d' can be determined using plane geometry in the $\vec{X}$-$\vec{W}$ plane shown in Figure 5.9.

Referring back to Figure 5.9 an expression for the angle α is derived first.

$$\alpha = \sin^{-1}\left(\frac{d/2}{r}\right)$$

Now r' is computed.

$$\frac{r'}{r} = \sin 2\alpha$$

$$\begin{aligned} r' &= r \sin\left(2\sin^{-1}\frac{d}{2r}\right) \\ &= 2r \sin\left(\sin^{-1}\frac{d}{2r}\right)\cos\left(\sin^{-1}\frac{d}{2r}\right) \\ &= 2r\,\frac{d}{2r}\sqrt{1-\left(\frac{d}{2r}\right)^2} \\ &= d\sqrt{1-\left(\frac{d}{2r}\right)^2} \end{aligned}$$

An intermediate result, s, is now derived.

$$\frac{s}{r} = \cos 2\alpha$$

$$\begin{aligned} s &= r\cos\left(2\sin^{-1}\frac{d}{2r}\right) \\ &= r\left(1-2\sin^2\left(\sin^{-1}\frac{d}{2r}\right)\right) \\ &= r\left(1-\frac{d^2}{2r^2}\right) \\ &= \frac{2r^2-d^2}{2r} \end{aligned}$$

With an expression for s, an expression for d' is derived.

$$\frac{d'}{s} = \tan\beta$$

$$d' = \frac{2r^2 - d^2}{2r} \tan\beta$$

Now an expression for γ is derived by referring to Figure 5.10.

$$\begin{aligned}
\gamma &= \cos^{-1}\frac{d'}{r'} \\
&= \cos^{-1}\left(\frac{2r^2 - d^2}{2r}\frac{\tan\beta}{d\sqrt{1-(\frac{d}{2r})^2}}\right) \\
&= \cos^{-1}\left(\frac{(2r^2 - d^2)\tan\beta}{d\sqrt{4r^2 - d^2}}\right) \\
&= \cos^{-1}(\rho\tan\beta) \qquad (5.13)
\end{aligned}$$

where ρ is defined,

$$\rho = \frac{2r^2 - d^2}{d\sqrt{4r^2 - d^2}}. \qquad (5.14)$$

The area of the neighbor hypersphere on the opposite side of the decision hyperplane from the true input vector is expressed by the integral,

$$\int_0^{\gamma} \mathrm{K}_{n-1}\ (r'\sin\phi)^{n-2}\ r'd\phi\,.$$

This integral sums up slices perpendicular to the line that d' lies on in Figure 5.10. Each slice is the area of a hypersphere of dimension $n-2$, and has a thickness of $r'\,d\phi$, where ϕ is an angle measured from the line d' lies on. Therefore, the probability of error when a neighbor pattern is presented instead of the true pattern is the ratio of the integral above to the total surface content of the neighbor pattern hypersphere,

P[decision error|input has error of magnitude d] =

$$\frac{\mathrm{K}_{n-1}}{\mathrm{K}_n}\int_0^{\cos^{-1}(\rho\beta)} \sin^{n-2}\phi\, d\phi \qquad (5.15)$$

This conditional probability is a function of β, the angle from $\vec{\mathrm{X}}$ to the decision hyperplane. For $\beta \approx 0$, the pattern $\vec{\mathrm{X}}$ lies very close to the decision hyperplane and the probability of an error would be one-half. Indeed, it can be verified that Equation 5.15 yields a value of one-half for $\beta = 0$. For $\beta > 2\alpha$, $\vec{\mathrm{X}}$ is sufficiently far from the decision hyperplane that none of the pattern neighbors produce errors. To get an average probability of error when the input is a pattern at distance d from the true input, the error rate above must be weighted by the probability density of finding the true pattern at angle β from the decision hyperplane.

For positive patterns, β is $\pi/2$ less the angle between $\vec{X}$ and $\vec{W}$. Using a modification of Equation 5.7 and remembering that the patterns exist on a hemihypersphere, the average probability of error for a single Adaline when the erroneous input is at a distance d from the true input can be written,

Ave P[decision error|input has error of magnitude d]

$$= \int_0^{2\alpha} \int_0^{\cos^{-1}(\rho\beta)} \frac{K_{n-1}}{K_n} \sin^{n-2}\phi \, d\phi \, \frac{K_n}{\frac{1}{2}K_{n+1}} \cos^{n-1}\beta \, d\beta$$

$$= 2\frac{K_{n-1}}{K_{n+1}} \int_0^{2\sin^{-1}\frac{d}{2r}} \int_0^{\cos^{-1}(\rho\beta)} \sin^{n-2}\phi \cos^{n-1}\beta \, d\phi \, d\beta \qquad (5.16)$$

This rather formidable double integral is the desired result. It can be evaluated numerically on a computer. The dependency of the inner integral's upper limit on the outer integral's variable of integration and the large range of integration of the inner integral, make it difficult to arrive at a closed form solution. The author's attempts to make approximations in order to continue an analytic reduction of the result to a more useful form have been unsuccessful. To within the limits set by the initial assumptions, the result above is rigorous but, unfortunately, difficult to use.

In an attempt to find an "alternate expression" that closely approximates the above result, the double integral was evaluated for many different cases. An expression that provides a very close fit to the rigorous result is given by,

$$\frac{1}{\pi}\frac{d}{r}. \qquad (5.17)$$

Remember that d is the magnitude of the input error, $|\Delta\vec{X}|$, which is $\sqrt{4h}$ for the binary case where h bits are in error. Also, r in the analysis above is the magnitude of the input vector, $|\vec{X}| = \sqrt{n+1}$, where n is the dimension of the input pattern. Using these facts, one realizes that the result given by the alternate expression is very similar to the result obtained hen decision errors due to weight errors were analyzed. Substitution into the alternate expression gives:

$$\frac{1}{\pi}\frac{|\Delta\vec{X}|}{|\vec{X}|}$$

The alternate expression is now very appealing to one's intuition. There exists a basic symmetry when working in the $(n+1)$-dimensional pattern/weight space. The defining equation for the Adaline's decision surface, $y = \vec{X}^T\vec{W} = \vec{W}^T\vec{X} = 0$, is symmetric in $\vec{X}$ and $\vec{W}$. It seems intuitive then, that weight errors and input errors would have similar effects

on the input/output mapping of the Adaline. The alternate expression is most accurate for small numbers of errors in the input patterns but remains useful for numbers of errors approaching half the number of inputs. The rigorously derived double integral expression cannot be expected to hold for errors numbering more than this due to the assumption about "edge effects" not affecting the distribution of neighbor patterns about the true pattern. The results section shows that the alternate expression in Equation 5.17 is accurate over a wide range of parameters.

5.5 Total Decision Errors for Adalines and Madalines

The results from the above two sections allow a formulation of the total probability of an Adaline making a decision error due to the combined effects of weight errors and input errors. In networks where it can be assumed that the Adalines are independent of each other, this result can be extended to predict the error rate of the entire network. To begin, the single Adaline is considered.

Consider a single Adaline that is subject to the combined effects of weight disturbances and binary input errors. Section 5.3 provides a result to predict the rate of decision errors when weight errors are present but there are no input errors. The results of Section 5.4 predict the decision error rate when patterns with a known number of errors in them are presented to an Adaline that has no weight errors. How do these results combine? It is argued here that when small weight errors are present, the above result for input errors is unaffected by weight errors that might be present.

The input error result shows the effect of using a pattern's neighbors at a given distance away from it as inputs to the Adaline instead of the pattern itself. For the unperturbed (no weight errors present) position of the decision hyperplane, there will be a set of patterns whose neighbors will cause errors if the neighbors are presented. This set of patterns is situated in a band on the pattern hypersphere. This band is centered on the intersection of the decision hyperplane with the pattern hypersphere. A slight movement of the hyperplane, due to weight errors, changes the set of patterns involved in decision errors, relative to the new position of the decision hyperplane. However, the number of such patterns involved in decision errors will remain the same. Some neighbor patterns, located near the edge of the affected band of patterns that caused decision errors for the unperturbed Adaline, will not cause errors in the weights-perturbed Adaline because the decision hyperplane has moved away from the neighbor. Similarly, some neighbors that didn't make errors before the weight

perturbation, will cause errors after the weights are perturbed. These two situations will offset each other.

Consider now the true patterns whose classifications change when the decision hyperplane shifts due to weight perturbations. These classification-affected patterns were close to the unperturbed decision hyperplane and are still close to the perturbed decision hyperplane, if the weight perturbation is small. It was noted in the previous section that the decision error rate due to input errors is one-half when the true pattern is near the decision hyperplane. This is true regardless of which side of the decision hyperplane the true pattern lies. A reversal of which neighbor patterns provide correct and incorrect responses occurs with the shift of the hyperplane for those true patterns that change classification. Again though, the effects offset each other. Thus, no net change in the rate of decision errors due to input errors occurs because of the introduction of the weight errors.

In summary, the following statements will be true when weight errors in the single Adaline are small. Errors in the weights will, on average, affect only those patterns which are presented with no input errors. The decision error rate for patterns presented with input errors, is unaffected by weight errors that are present.

An introduction of new notation facilitates writing the results compactly. Define $\mathcal{D}_h$ as the probability that an Adaline makes a decision error given that its input has h errors. Define $\mathcal{E}$ as the total probability that an Adaline makes a decision error. Then,

$$\mathcal{E} = \sum_{h=0}^{n} \mathcal{D}_h \cdot \mathrm{P}[\text{input has } h \text{ errors}] \tag{5.18}$$

From the discussion above about the independence of weight errors and input errors it is concluded that,

$$\mathcal{D}_0 \approx \frac{1}{\pi}\frac{|\Delta\vec{\mathrm{W}}|}{|\vec{\mathrm{W}}|}\left(\frac{\mathrm{K}_n}{\mathrm{K}_{n+1}}\right)^2\frac{2\pi}{n} \tag{5.19}$$

$$\approx \frac{1}{\pi}\frac{|\Delta\vec{\mathrm{W}}|}{|\vec{\mathrm{W}}|} \tag{5.20}$$

These are the results for weight errors without input errors from Section 5.3. By Equation 5.18 though, these results affect only that fraction of the input patterns presented without errors. $\mathcal{D}_h$ for h other than zero is obtained from Equation 5.16 or 5.17. At this point use the facts that $r = \sqrt{n+1}$ and $d = \sqrt{4h}$ in these two equations and the defining

equation for ρ. As explicit functions of h and n the expressions become,

$$\mathcal{D}_h = 2\frac{K_{n-1}}{K_{n+1}}\int_0^{2\sin^{-1}\sqrt{\frac{h}{n+1}}}\int_0^{\cos^{-1}(\rho\beta)}\sin^{n-2}\phi\,\cos^{n-1}\beta\,d\phi\,d\beta \quad , h \neq 0 \tag{5.21}$$

$$\approx \frac{1}{\pi}\sqrt{\frac{4h}{n+1}} \quad , h \neq 0 \tag{5.22}$$

and ρ as defined in Equation 5.14 is now given by,

$$\rho = \frac{n+1-2h}{2\sqrt{h(n+1-h)}} \tag{5.23}$$

The equations above allow one to predict the average decision errors that a single Adaline with perturbed weights and input errors will make relative to its unperturbed reference which receives inputs without error. The use of Equations 5.19 and 5.21 in Equation 5.18 is referred to as the complicated approximation. The complicated approximation will provide the best prediction of decision error rate that the rigorous analyses can provide. It is an approximation for two reasons. During the analyses, some analytic approximations were made, such as $\tan x \approx x$, for small x. Secondly, it has often been assumed that weight perturbations and number of input errors would be small. The argument about weight errors not affecting the decision error rate for input errors hinges on the small weight error assumption. The expressions used in the complicated approximation can be evaluated using a computer. A simple approximation is presented that allows evaluation using paper and pencil. The simple approximation will mean the use of the simpler expressions of Equations 5.20 and 5.22 in Equation 5.18. The simple approximation is not analytically rigorous since it uses an alternate expression that provides a "good fit" to an analytic result. The simple approximation's merit is its ease of use. The results, in the next section, contrast the predictions of the complicated and simple approximations and compare the predictions with simulation results.

The results above are now extended to Madalines in which the Adaline output errors occur independently of each other. An analysis begins with the first layer. If only weight errors are present, Equation 5.19 or 5.20, depending upon whether one uses the complicated or simple approximation, provides the error rate for each Adaline on the first layer. The binomial distribution of Equation 5.12 is used to predict the probability of the first hidden pattern having a given number of errors. This then is the input error distribution for the second-layer Adalines. This information is used in Equation 5.18 to predict the error rate

for each Adaline in the second layer, etc. For a Madaline with l layers, the decision error rate of an Adaline on the ith layer, denoted $\mathcal{E}_i$, is given by:

$$\mathcal{E}_i = \sum_{h=0}^{n} \mathcal{D}_{ih} \cdot \text{P[input pattern has } h \text{ errors]} \qquad i = 1 \tag{5.24}$$

$$= \sum_{h=0}^{n_{i-1}} \mathcal{D}_{ih} \cdot \text{P[input to layer } i \text{ has } h \text{ errors]} \qquad i = 2, \ldots, l$$

$$= \sum_{h=0}^{n_{i-1}} \mathcal{D}_{ih} \frac{n_{i-1}!}{(n_{i-1}-h)!\,h!} \mathcal{E}_{i-1}^{h} (1-\mathcal{E}_{i-1})^{n_{i-1}-h} \qquad i = 2, \ldots, l \tag{5.25}$$

The last expression specifically includes the binomial distribution for the errors in the input patterns to layer i. The added subscript, i, on the $\mathcal{D}_h$s indicates that the formulas for $\mathcal{D}_h$ must be adjusted for the number of inputs to the ith layer. Finally, $\mathcal{E}_l$ can be used to predict the output error rate of the entire Madaline. For a Madaline with n_l output units, the output pattern will have one or more errors in it at the rate given by:

$$\mathcal{E}_{\text{output}} = 1 - (1 - \mathcal{E}_l)^{n_l} \tag{5.26}$$

When comparing a perturbed net against a reference, as in Figure 5.1, one must assume the input patterns to the first layer contain no errors. The reference network responds to what it sees as if it were the true input. The perturbed net's response is compared against the reference network's output with the implicit assumption that the reference network's response is correct. For a meaningful comparison, the input, which both networks receive in parallel, must not have errors in it. Useful Madalines will stand alone. They won't be compared against a reference network in real applications. Such Madalines may operate in an environment where the input patterns have errors in them. As explicitly provided for in Equation 5.24, the theory developed above has the power to use knowledge about the distribution of input errors when predicting the total decision error rate for real-world applications where errors are present in the input patterns.

5.6 Simulation Results

The system depicted in Figure 5.1 is used to test the theory developed in the previous sections. A reference Madaline system, with a particular architecture, is generated randomly by selecting each weight in the system independently from a uniform distribution over the

interval $(-1,+1)$. These weights are then copied to the perturbed system which has an architecture identical to the reference system.

To the weight vectors of each Adaline in the perturbed system a weight perturbation vector $\Delta\vec{W}$ is added. This vector is generated by selecting each component independently from the same uniform distribution as before. The perturbation vector is then scaled so that a desired weight perturbation ratio, $|\Delta\vec{W}|/|\vec{W}|$, is obtained. Thus, the assumptions of the derivation are maintained. The orientation of the perturbation weight vector is random and independent of the reference weight vector but has a fixed magnitude relative to the reference. This weight perturbation ratio is expressed as a percentage in the following results.

A large number of patterns from the input space are then presented to the reference and perturbed systems in parallel, and their outputs compared. If any bit of the outputs differ, a decision error is said to be made. The results present the number of decision errors as a percentage of the patterns tested. For any of the given architectures being checked, the procedure is repeated for several different reference systems i.e., several sets of fixed random network weights. Several perturbations of each reference system are tested. The results presented are the average percent decision errors over the different references and their different perturbations.

The first system investigated is that of a single Adaline. No input errors are allowed. Random perturbations are inserted into the weights. The simulation uses fifty reference Adalines. Each reference is compared with fifty different perturbations of itself and the results averaged. This is repeated at each value of weight perturbation ratio. The predictions by the simple and complicated theory approximations and the simulation averages are tabulated for different n and $|\Delta\vec{W}|/|\vec{W}|$ in Table 5.1.

The results show there is a weak dependence on n which is well predicted by the complicated theory approximation. The simple approximation is good in the limit as n grows large and gives an upper limit on the decision errors. The simulation results generally lie between the predictions of the two formulas. To gain an appreciation of the weak dependence on n and how close the simple approximation is, the simulation data and the simple approximation predictions are plotted in Figure 5.11. The plots nearly lie on top of each other. The complicated approximation would lie on top of the simulation plots and is not shown. The important thing to note is the linear dependence of percent decision errors on the weight perturbation ratio as predicted by the theory and borne out by the simulation results. It is concluded that the theory is very accurate in predicting the effects of weight

Single Adaline Sensitivity Study			
$\frac{\lvert\Delta\vec{W}\rvert}{\lvert\vec{W}\rvert}$ (%)	% Decision Errors Simple Approximation	% Decision Errors Complicated Approximation	% Decision Errors Simulation Results
$n = 8$			
5	1.59	1.50	1.53
10	3.18	2.99	3.11
20	6.37	5.98	6.15
30	9.55	8.97	9.06
$n = 16$			
5	1.59	1.54	1.56
10	3.18	3.09	3.12
20	6.37	6.17	6.19
30	9.55	9.26	9.17
$n = 30$			
5	1.59	1.57	1.57
10	3.18	3.13	3.14
20	6.37	6.26	6.24
30	9.55	9.39	9.22
$n = 49$			
5	1.59	1.58	1.58
10	3.18	3.15	3.16
20	6.37	6.30	6.25
30	9.55	9.45	9.24

Table 5.1: Effects of perturbing the weights of a single Adaline. The predicted and simulated percentage decision errors for weight disturbance ratios of 5, 10, 20 and 30 percent are shown for Adalines with 8, 16, 30, and 49 inputs.

perturbations on a single Adaline.

The simulation results presented above validate the formulas used in both the simple and complicated approximations for $\mathcal{D}_0$. Next, a Madaline architecture with many first-layer Adalines and a single output Adaline is simulated. This simulation allows a check of the formulas for $\mathcal{D}_h$, $h \neq 0$. During the simulation, the output of the first layer of the perturbed network is compared with the output of the first layer of the reference network and the number of errors between the two is noted. Then it is noted whether the decision rendered by the perturbed system is the same as that of the reference. This allows computation of

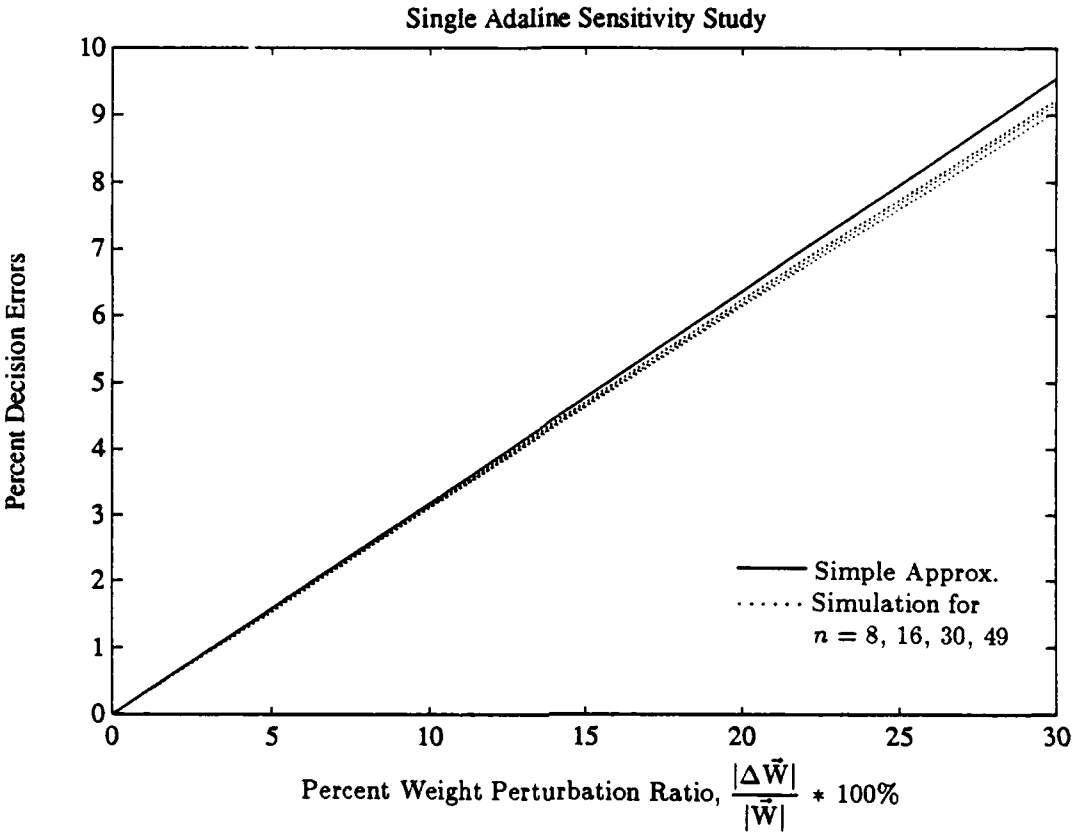


Figure 5.11: Sensitivity of the single Adaline. Dotted lines are simulation results for Adalines with 8, 16, 30, and 49 inputs. Solid line is the theory prediction by the simple approximation. Data is taken from Table 5.1.

the rate of decision errors due to the input pattern having a particular number of errors.

Reference and perturbed Madalines with 49 inputs, 35 first-layer Adalines and 1 output Adaline are simulated for testing in the configuration of Figure 5.1. The weight perturbation ratio is 20% for each Adaline in the perturbed system. The error rate for each perturbed first-layer Adaline is just $\mathcal{D}_0$ since the inputs have no errors. This rate is used in the binomial distribution to predict the distribution of input errors to the second-layer Adaline of the perturbed system. Since the simple and complicated theory approximations use different formulae for $\mathcal{D}_0$, the predicted frequency of each number of input errors to the output Adaline will be slightly different for the two approximations. The rates at which the second-layer (output) Adaline gives an erroneous output to its input pattern containing a given number of errors are given by the formulas for $\mathcal{D}_h$ with the understanding that the output Adaline has 35 inputs. The total error rate for the output Adaline is computed

Input Errors and Decision Errors for Output Adaline 49 input, 35-feed-1 Madaline, $\frac{\lvert\Delta W\rvert}{\lvert W\rvert} = .2$						
	frequency (%) of input errors			$\mathcal{D}_h$ (%), for inputs with h errors		
errors	simple	complicated	observed	simple	complicated	observed
0	10.00	10.25	10.47	6.36	6.28	6.13
1	23.80	24.12	24.41	10.61	10.66	12.78
2	27.51	27.58	27.66	15.00	15.15	16.48
3	20.58	20.40	20.23	18.38	18.64	19.66
4	11.19	10.98	10.78	21.22	21.63	22.48
5	4.72	4.58	4.45	23.72	24.31	24.96
6	1.60	1.54	1.48	25.99	26.77	27.26
7	.45	.43	.40	28.07	29.07	29.52

$\mathcal{E}_{\text{simple}} = 15.15\%$

$\mathcal{E}_{\text{complicated}} = 15.26\%$

$\mathcal{E}_{\text{observed}} = 16.39\%$

Table 5.2: An output Adaline sees errors in its input relative to the reference network. Predicted and observed frequency of each number of input errors and predicted and observed error rates for each number of input errors are presented.

using Equation 5.18. All of this information for both the complicated and simple theoretical approximations as well as simulation results is presented in Table 5.2.

The table shows excellent agreement between the simple and complicated theory approximations. Of particular interest, is the agreement between the simple and complicated theory approximations for $\mathcal{D}_h$, $h \neq 0$. Remember that the simple approximation uses an alternate expression found empirically for the rigorous result derived in Section 5.4. The results presented here involve weight perturbations of 20%. At this very high level of disturbance, the simple and complicated theories underestimate the observed error rates for inputs with one error or more. The theories are closer at lower disturbance levels but remain useful at even higher weight perturbation ratios.

The next result shows how well the theory predicts output Adaline error rates over a range of weight perturbation ratios. The simulation system is an n-input n_1-feed-1 network. A typical result is presented for $n = n_1 = 16$. In this simulation, 15 reference nets are perturbed 25 times for each value of the weight perturbation ratio. The average results are tabulated in Table 5.3 and plotted in Figure 5.12.

The simple and complicated theoretical approximations provide predictions that are

Single-output Madaline Sensitivity Study			
$\frac{\lvert\Delta\vec{W}\rvert}{\lvert\vec{W}\rvert}$ (%)	% Decision Errors Simple Approximation	% Decision Errors Complicated Approximation	% Decision Errors Simulation Results
10	8.79	8.66	8.97
20	14.46	14.39	14.96
30	18.45	18.49	19.47

Table 5.3: Percent decision errors for a 16-input, 16-feed-1 Madaline with weight perturbation ratios of 10, 20, and 30 percent.

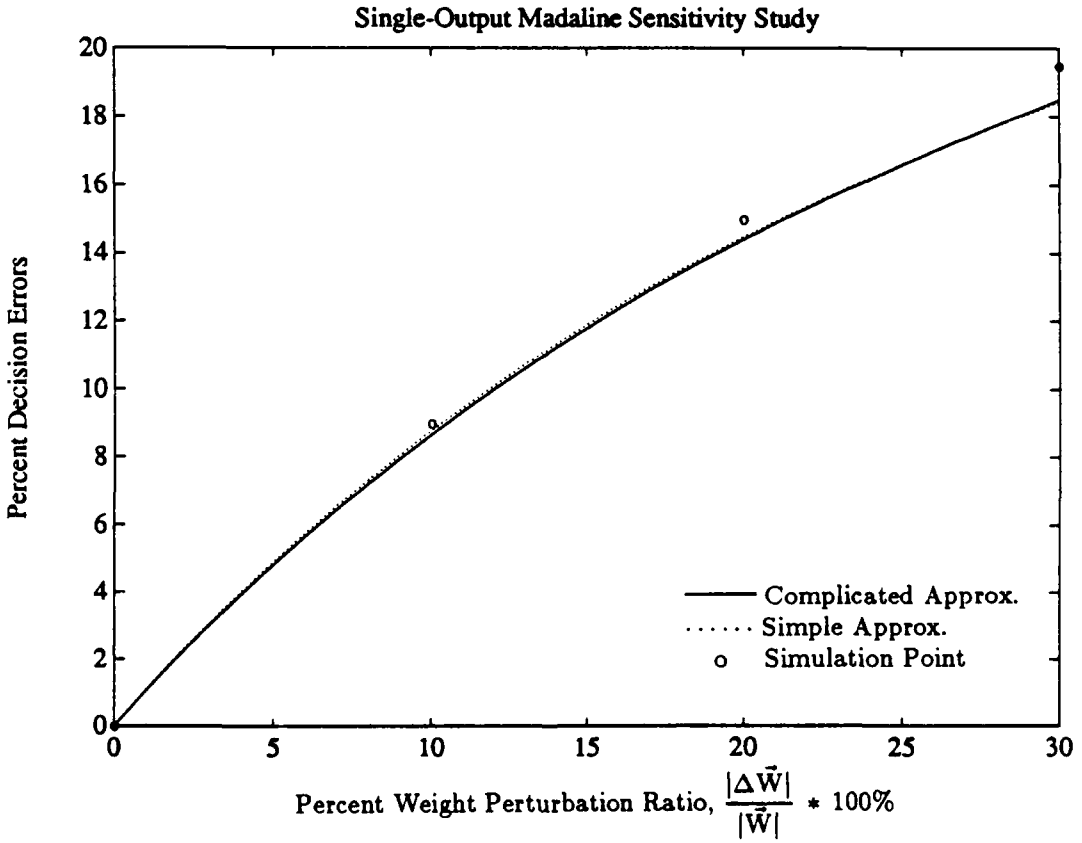


Figure 5.12: Percent decision errors versus percent weight perturbation. Network is a 16-input, 16-feed-1 Madaline. Theory predictions by two different levels of approximation are shown with simulation results. Data from Table 5.3.

Multioutput Madaline Sensitivity Study			
$\frac{\lvert\Delta\vec{W}\rvert}{\lvert\vec{W}\rvert}$ (%)	% Decision Errors Simple Approximation	% Decision Errors Complicated Approximation	% Decision Errors Simulation Results
1	3.75	3.73	3.60
5	15.69	15.64	15.44
10	25.93	25.93	26.11
20	38.42	38.65	39.95
30	46.17	46.66	49.12

Table 5.4: Simulation vs. theory for a multioutput Madaline. The network is a 49-input, 25-feed-3. A decision error occurs when any bit of the output is different from the reference.

nearly identical over the whole range of perturbations. The predictions and the simulation results are acceptably close though diverging somewhat at the higher perturbation levels. This is not surprising. The analyses assumed small weight perturbations. The curves in Figure 5.12 indicate the limits of usefulness, in terms of weight perturbation ratio, of the theoretical results. Simulation results begin to deviate significantly from the theoretical predictions for weight perturbation ratios in excess of 20%.

To test the theories on a multi-output network, a 49-input 25-feed-3 network is simulated. Ten reference nets, each perturbed 15 times for each weight perturbation ratio tested, are averaged to get the results in Table 5.4. The theoretical predictions come from Equation 5.26 where the simple and complicated approximation expressions are used in Equations 5.24 and 5.25 to get $\mathcal{E}_2$ ($l = 2$ in this case). Here, weight perturbation ratios lower than any previously presented are used to check the accuracy of the predictions given by the simple and complicated expressions at smaller disturbance levels. A plot of the simulation data alongside the simple and complicated theory approximations is shown in Figure 5.13. The excellent agreement between the two approximations of the theory is repeated in this example. Both provide very accurate predictions compared to the simulation results.

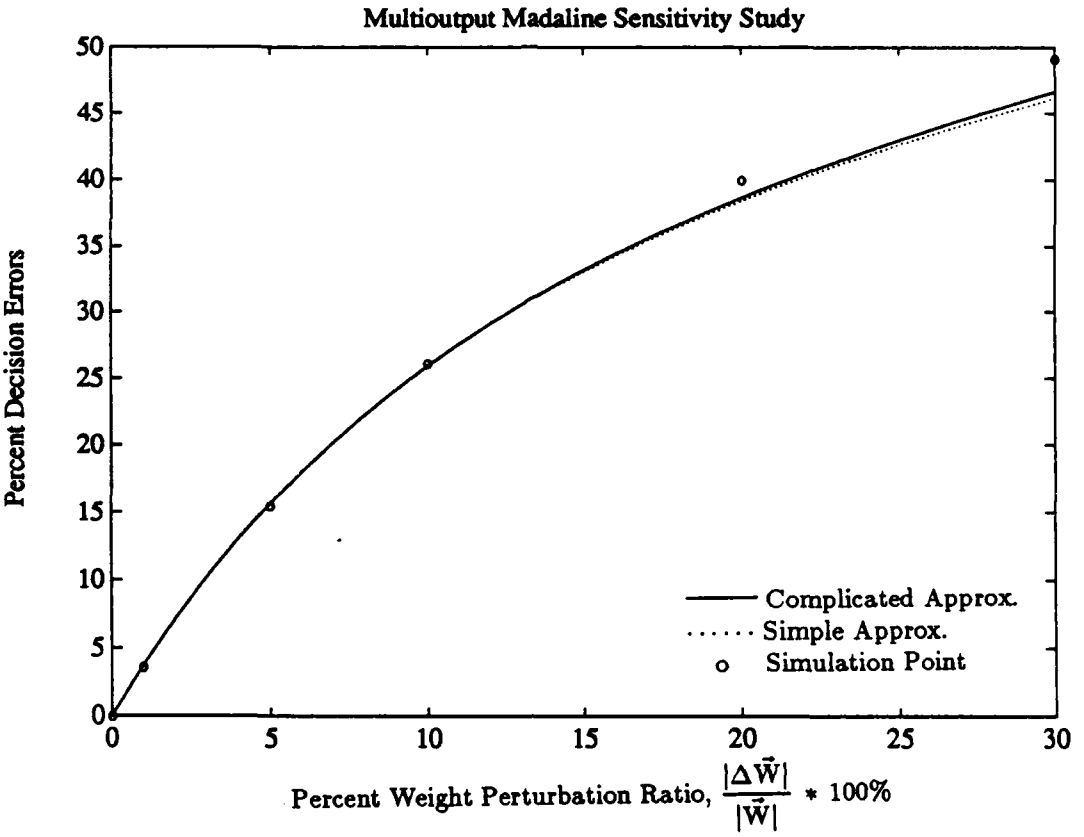


Figure 5.13: Decision Errors vs. Weight Perturbation Ratio for a multioutput Madaline. Network is 49-input, 25-feed-3. Data taken from Table 5.4.

5.7 Discussion

This chapter derives a fairly simple theory. The theory relates the change of the input/output mapping of a Madaline to the size of weight perturbations relative to a reference system. Two different levels of approximation of the theory are addressed throughout the chapter. The approximations do not differ from each other by very much and closely predict operating results obtained by simulation. The results obtained by using the simple expressions for $\mathcal{D}_0$ and $\mathcal{D}_h$, $h \neq 0$, in the theory make the added computation of the more complicated expressions unnecessary.

The basic theory is valid for the single Adaline and is derived using two assumptions. It is assumed that the weight perturbations are independent of the reference weights and that knowledge of the ratio of the magnitude of the perturbation weight vector to the magnitude of the reference weight vector is known. The second assumption is that the distribution of input errors is known or computable and that errors occur independently by bit.

This theory for the single Adaline is applicable to networks of Adalines if the reference Adalines have independently chosen random fixed weights, and the perturbations of the weights in the perturbed network are all independent. The motivation for developing the theory was to gain an appreciation of how close to the reference solution a trained solution must be to get good output correspondence. The theory gives some idea of the effects of errors in the weights upon the errors of the output of a layered, trainable network.

This issue of closeness can now be addressed. One measure of closeness between two weight vectors is the angle between them. The most basic result derived in this chapter, Equation 5.6, says performance degradation is directly proportional to this angle. Another result shows this angle is the same as the weight perturbation ratio when n is large (for small angle). A 10 percent weight perturbation ratio means the angle between the two weight vectors being compared is about 6 degrees (.1 radian). A single Adaline, with no input errors to aggravate the situation, suffers a performance difference of about 3% ($10\%/\pi$). The performance degradation is also linear in the deviation angle or equivalently, the weight perturbation ratio. An angle of 12 degrees between two weight vectors causes a 6% difference in the mappings of the two Adalines.

Useful networks often require more than one Adaline. The binomial distribution is unforgiving. Suppose a network's first layer has 6 Adalines each of which respond correctly to 94% of its inputs. The aggregate response of the layer will be completely correct only about 69% of the time if the Adalines make their errors independently of each other. If this layer is providing inputs to another layer, this is disastrous. Or is it?

Remember that the error rate of a single Adaline due to weight disturbances is only weakly dependent on the number of inputs and is actually upper bounded by the simple theory approximation. How is the error rate of a second layer Adaline affected by the number of Adalines on the first layer? For the example shown in Table 5.2, about 90% of the hidden patterns presented to the output Adaline have an error in them. Only 15% of the output Adaline's responses are incorrect. Thus, the output Adaline "cleans up" most of the errors presented to it. How is this ability affected by the number of Adalines on the first layer for a given weight perturbation ratio?

Theory predictions and simulation results are used to investigate this matter. The results are shown in Figure 5.14. The networks all have 49 inputs and the same percent weight perturbation ratio of 20%. Networks with n_1 = 8, 16, 25, 35 and 49 are simulated. For n_1 greater than about 15, theory predicts and simulation bears out that output error rate is unaffected by the number of first-layer units. This is similar to the single Adaline

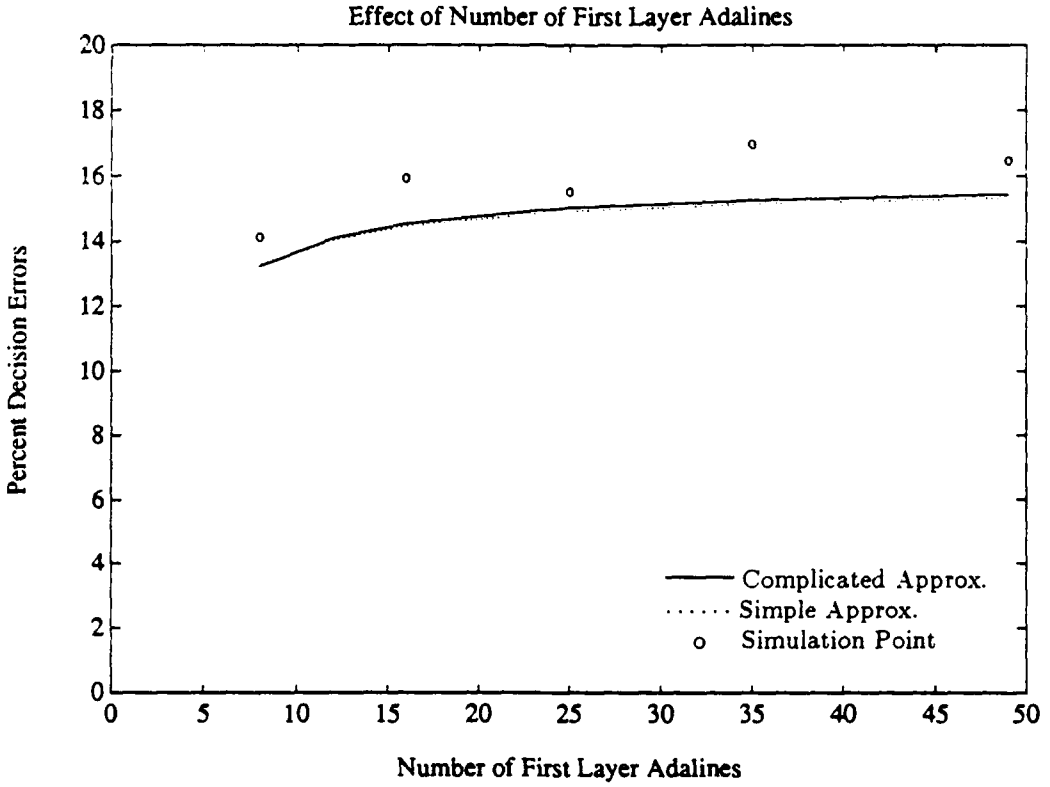


Figure 5.14: Sensitivity of networks as n_1 is varied. The net has 49 inputs, 1 output Adaline and the percent weight perturbation ratio is 20%.

with no input errors.

The error rate per output Adaline is somewhat immune then to the architecture of the network preceding it. The true curse of the binomial distribution is felt at the output layer. The ability of multi-output Madalines to produce a response with all bits correct is very dependent on how many output Adalines there are. A small per Adaline error rate in the output layer can cause large output pattern error rates in systems with only a few output Adalines. The 3-output Madaline whose performance is graphed in Figure 5.13, has decision errors in 25% of its responses for only a ten percent weight perturbation error.

This sensitivity shown by randomly generated networks indicate they are difficult to emulate. To achieve good training performance with the fixed/adaptive emulator system of the type of Chapter 4 requires very precise learning by the adaptive network. It becomes understandable that the MRII algorithm has difficulty learning the mappings presented by these random nets. They are hard problems from a sensitivity point of view.

To summarize the chapter, the complete theory that predicts the decision error rate of a weights-perturbed Madaline relative to an unperturbed reference is given. The simple expressions for $\mathcal{D}_0$ and $\mathcal{D}_h$, $h \neq 0$, are used as they have been found to be adequate in providing good predictions. To allow a more compact formulation, it will be understood that n_0 is the number of components in the input pattern. The assumptions made earlier about the independence of individual Adalines and independence of the weight perturbations must be valid. The formulation presented here allows errors in the input patterns to the perturbed Madaline, but the reference system must receive the true input. The theory for a Madaline with l layers:

$$\mathcal{E}_1 = \sum_{h=0}^{n} \mathcal{D}_{1h} \cdot \text{P[input has } h \text{ errors]}$$

$$\mathcal{E}_i = \sum_{h=0}^{n_{i-1}} \mathcal{D}_{ih} \cdot \frac{n_{i-1}!}{h!(n_{i-1}-h)!} \mathcal{E}_{i-1}^{h} \, (1-\mathcal{E}_{i-1})^{n_{i-1}-h} \qquad \text{for } i = 2, \ldots, l$$

where,

$$\mathcal{D}_{ih} = \frac{1}{\pi} \frac{|\Delta \vec{W}|}{|\vec{W}|} \qquad \text{for } h = 0$$

$$= \frac{1}{\pi} \sqrt{\frac{4h}{n_{i-1}+1}} \qquad \text{for } h = 1, \ldots, n_{i-1}$$

Finally, the output decision error rate of the entire Madaline is given by:

$$\mathcal{E}_{\text{output}} = 1 - (1 - \mathcal{E}_l)^{n_l}$$

Chapter 6

Conclusions

6.1 Summary

Madaline Rule II is based on the principle of minimal disturbance. Modifications to the principle are required. The issue of circuit implementation of the algorithm forces compromises in the number and types of trial adaptations that are performed. Minimal disturbance also leads to a local minima phenomenon that requires the implementation of a concept called usage. Usage implements a concept of responsibility sharing by forcing the participation of all hidden units in arriving at a hidden pattern representation of the input patterns.

MRII, as presented in this report, suffers from another kind of local minima phenomenon that takes the form of a limit cycle. Sometimes, certain output units arrive at a solution to their part of the global problem ahead of the rest of the output units. The hidden set determined at this point in the training is not separable by the "unsolved" output units in the manner they require. The output units that have come to a solution using this hidden pattern set, prevent further modification of the hidden pattern set. Any changes made to the hidden pattern set destroy the solutions arrived at by the "solved" output units. MRII will not allow this destruction to occur. This situation is a most difficult one. A departure from the principles of minimal disturbance is required but it is difficult to know precisely how to depart. A simple escape from this failure situation reinitializes the network with random weights and begins training again. This is wasteful of the previous training performed. The idea of using random changes in the weights to escape this failure was conceived. This led to a study of the sensitivity of the input/output mapping of a random Adaline to weight changes and input errors.

The sensitivity analysis yielded a simple set of formulas. They predict quite well how the mapping of an Adaline with randomly generated weights will be affected by weight changes and input errors. In nets where the weights of the units are set independently of each other, the results can be applied to predict the sensitivity of the entire net. Such networks are found to be quite sensitive to changes in their weights. The input/output mappings that random networks generate can be very challenging for an adaptive network to learn.

In spite of failure modes and being tasked by difficult problems, MRII exhibits some useful performance. It has the ability to train networks to perform a wide variety of tasks. It also exhibits interesting generalization properties. In networks trained by MRII, training and generalization performance track each other very closely. Generalization is also found to be robust in networks that have many more units than the minimum necessary to solve a given problem. This applies when there is a sufficiently large training set available. The use of oversized nets with large numbers of training patterns can lead to impressive improvements in training performance without suffering any significant degradation of generalization performance.

6.2 Suggestions for Further Research

The results of the sensitivity study have not been applied to the MRII failure mode. It was hoped the findings would allow a more thoughtful approach to using random noise in the weights for escaping the apparent limit cycles and local minima that MRII encounters. Further work in this area needs to be done.

The generalization properties of MRII should be explored further. The ability to maintain good generalization when using networks larger than required is very significant. No specific mechanism for insuring this is included in MRII. An investigation of why generalization is maintained might yield some useful information. The geometric distances between hidden patterns mapped to the same output pattern versus those mapped to other outputs should be checked. A related point is how sensitivity is affected by overarchitecturing.

The results of research into these areas would be useful in their own right. The results might also suggest improvements to the MRII algorithm. The author intends to pursue these issues as time permits, but challenges his colleagues to pursue them also.

Bibliography

[1] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," in *IRE WESCON Conv. Rec., pt. 4*, pp. 96–104, 1960.

[2] B. Widrow, "Generalization and information storage in networks of adaline 'neurons'," in *Self Organizing Systems 1962*, (M. C. Yovitz, G. T. Jacobi, and G. D. Goldstein, eds.), pp. 435–461, Washington, DC: Spartan Books, 1962.

[3] F. H. Glanz, "Statistical extrapolation in certain adaptive pattern-recognition systems," Tech. Rep. 6767-1, Stanford Electronics Laboratories, Stanford, CA, May 1965.

[4] B. Widrow, "An adaptive "adaline" neuron using chemical "memistors"," Tech. Rep. 1553-2, Stanford Electronics Laboratories, Stanford, CA, Oct. 1960.

[5] C. H. Mays, "Adaptive threshold logic," Tech. Rep. 1557-1, Stanford Electronics Laboratories, Stanford, CA, Apr. 1963.

[6] W. C. Ridgway III, "An adaptive logic system with generalizing properties," Tech. Rep. 1557-1, Stanford Electronics Laboratories, Stanford, CA, Apr. 1962.

[7] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing.* Vol. I and II, Cambridge, Massachusetts: MIT Press, 1986.

[8] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, no. 79, pp. 2554–2558, 1982.

[9] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann machines: constraint satisfaction networks that learn," Tech. Rep. CMU-CS-84-119, Carnegie-Mellon University, Department of Computer Science, Pittsburgh, PA, 1984.

[10] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1985.

[11] B. Widrow, R. G. Winter, and R. A. Baxter, "Learning phenomena in layered neural networks," in *IEEE First Annual International Conference on Neural Networks*, (San Diego, CA), 1987.

[12] M. E. Hoff, Jr., "Learning phenomena in networks of adaptive switching circuits," Tech. Rep. 1554-1, Stanford Electronics Laboratories, Stanford, CA, July 1962.

[13] R. J. Brown, "Adaptive multiple-output threshold systems and their storage capacities," Tech. Rep., Stanford Electronics Labs. Rep. 6671-1, Stanford University, Stanford CA, June 1964.

[14] M. G. Kendall, *A Course in the Geometry of n Dimensions.* London: Charles Griffin & Company Ltd., 1961.

[15] D. M. Y. Sommerville, *An Introduction to the Geometry of N Dimensions.* London: Methuen & Co. Ltd., 1929.

[16] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* New York, NY: McGraw-Hill, second ed., 1984.